```
FFFFFFFFFFFFFFF     AAAAAAAAA    LLL
FFFFFFFFFFFFFFF     AAAAAAAAA    LLL
FFFFFFFFFFFFFFF     AAAAAAAAA    LLL
FFF              AAA         AAA LLL
FFF              AAA         AAA LLL
FFF              AAA         AAA LLL
FFF              AAA         AAA LLL
FFF              AAA         AAA LLL
FFF              AAA         AAA LLL
FFFFFFFFFFFF     AAA         AAA LLL
FFFFFFFFFFFF     AAA         AAA LLL
FFFFFFFFFFFF     AAA         AAA LLL
FFF              AAAAAAAAAAAAAAA LLL
FFF              AAAAAAAAAAAAAAA LLL
FFF              AAAAAAAAAAAAAAA LLL
FFF              AAA         AAA LLL
FFF              AAA         AAA LLL
FFF              AAA         AAA LLL
FFF              AAA         AAA LLLLLLLLLLLLLLL
FFF              AAA         AAA LLLLLLLLLLLLLLL
FFF              AAA         AAA LLLLLLLLLLLLLLL
```

```
_$

Val
---
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
000
7FF
7FF
7FF
7FF
7FF
7FF
7FF
7FF
7FF
7FF
7FF
7FF
7FF
7FF
7FF
7FF
7FF
7FF
7FF
7FF
7FF
7FF
7FF
7FF
7FF
7FF
7FF
7FF
7FF
7FF
7FF
7FF
7FF
```

```
                                                                 B  3
**FILE**ID**FALDECODE





          FFFFFFFFFF    AAAAAA    LL          DDDDDDDD    EEEEEEEEEE    CCCCCCCC    OOOOOO    DDDDDDDD    EEEEEEEEEE
          FFFFFFFFFF    AAAAAA    LL          DDDDDDDD    EEEEEEEEEE    CCCCCCCC    OOOOOO    DDDDDDDD    EEEEEEEEEE
          FF          AA      AA  LL          DD      DD  EE          CC          OO      OO  DD      DD  EE
          FF          AA      AA  LL          DD      DD  EE          CC          OO      OO  DD      DD  EE
          FF          AA      AA  LL          DD      DD  EE          CC          OO      OO  DD      DD  EE
          FF          AA      AA  LL          DD      DD  EE          CC          OO      OO  DD      DD  EE
          FFFFFFFF    AA      AA  LL          DD      DD  EEEEEEEE    CC          OO      OO  DD      DD  EEEEEEEE
          FFFFFFFF    AA      AA  LL          DD      DD  EEEEEEEE    CC          OO      OO  DD      DD  EEEEEEEE
          FF          AAAAAAAAAA  LL          DD      DD  EE          CC          OO      OO  DD      DD  EE
          FF          AAAAAAAAAA  LL          DD      DD  EE          CC          OO      OO  DD      DD  EE
          FF          AA      AA  LL          DD      DD  EE          CC          OO      OO  DD      DD  EE              ....
          FF          AA      AA  LL          DD      DD  EE          CC          OO      OO  DD      DD  EE              ....
          FF          AA      AA  LLLLLLLLLL  DDDDDDDD    EEEEEEEEEE    CCCCCCCC    OOOOOO    DDDDDDDD    EEEEEEEEEE      ....
          FF          AA      AA  LLLLLLLLLL  DDDDDDDD    EEEEEEEEEE    CCCCCCCC    OOOOOO    DDDDDDDD    EEEEEEEEEE      ....

          LL            IIIIII      SSSSSSSS
          LL            IIIIII      SSSSSSSS
          LL              II      SS
          LL              II      SS
          LL              II      SS
          LL              II      SS
          LL              II        SSSSSS
          LL              II        SSSSSS
          LL              II              SS
          LL              II              SS
          LL              II              SS
          LL              II              SS
          LLLLLLLLLL    IIIIII    SSSSSSSS
          LLLLLLLLLL    IIIIII    SSSSSSSS
```

Table of contents

```
0000            .TITLE  FALDECODE - DECODE DAP MESSAGE
0000            .IDENT  'V04-000'
0000
0000    ;
0000    ;****************************************************************************
0000    ;*                                                                          *
0000    ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
0000    ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                  *
0000    ;*  ALL RIGHTS RESERVED.                                                    *
0000    ;*                                                                          *
0000    ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
0000    ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
0000    ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
0000    ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
0000    ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
0000    ;*  TRANSFERRED.                                                            *
0000    ;*                                                                          *
0000    ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
0000    ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
0000    ;*  CORPORATION.                                                            *
0000    ;*                                                                          *
0000    ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
0000    ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
0000    ;*                                                                          *
0000    ;*                                                                          *
0000    ;****************************************************************************
0000    ;
0000
0000    ;++
0000    ; Facility: FAL (DECnet File Access Listener)
0000    ;
0000    ; Abstract:
0000    ;
0000    ;       This module decodes (parses) the next DAP message and stores the
0000    ;       validated fields in the DAP control block.
0000    ;
0000    ; Environment: VAX/VMS, user mode
0000    ;
0000    ; Author: James A. Krycka,      Creation Date:  16-JUN-1977
0000    ;
0000    ; Modified By:
0000    ;
0000    ;       V03-008 JEJ0048         J E Johnson     13-Jul-1984
0000    ;               Eliminate the check for a file name of 128 characters or
0000    ;               more.
0000    ;
0000    ;       V03-007 JEJ0019         J E Johnson     27-Mar-1984
0000    ;               Alter CECK_OPERATING_SYSTEM to use DAP$V_P_OS as the
0000    ;               P/OS flag due to naming conflict with DAP$V_POS magtape
0000    ;               positioning flag. Also use DAP$K_P_OS.
0000    ;
0000    ;       V03-006 JAK0124         J A Krycka      06-SEP-1983
0000    ;               Update new DAP$Q_DCODE_FLG status bits during parse of
0000    ;               Configuration message.
0000    ;
0000    ;       V03-005 JAK0113         J A Krycka      22-JUN-1983
0000    ;               Continuation of support for DAP V7.0 spec.
```

```
0000     ;               Add support for 64-bit binary keys.
0000     ;               Also, set DAP$V_VMS_XPFn flags as appropriate.
0000     ;
0000     ;       V03-004 KRM0105         K Malik         10-May-1983
0000     ;               Update to support DAP V7.0 specification.
0000     ;
0000     ;       V03-003 KRM0085         K Malik         23-Mar-1983
0000     ;               Add support for STMLF and STMCR file formats.
0000     ;               Also, set DAP$V_GEQ_V70 bit as appropriate.
0000     ;
0000     ;       V03-002 KRM0069         K Malik         23-Nov-1982
0000     ;               Add support for $RENAME service.
0000     ;
0000     ;--
```

```
                   0000             .SBTTL  DECLARATIONS
                   0000
                   0000    ;
                   0000    ; Include Files:
                   0000    ;
                   0000
                   0000             $DAPPLGDEF                      ; Define DAP prologue symbols
                   0000             $DAPHDRDEF                      ; Define DAP message header
                   0000             $DAPSSPDEF                      ; Define DAP system specific field
                   0000             $DAPCNFDEF                      ; Define DAP Configuration message
                   0000             $DAPATTDEF                      ; Define DAP Attributes message
                   0000             $DAPACCDEF                      ; Define DAP Access message
                   0000             $DAPCTLDEF                      ; Define DAP Control message
                   0000             $DAPCONDEF                      ; Define DAP Continue Transfer message
                   0000             $DAPACKDEF                      ; Define DAP Acknowledge message
                   0000             $DAPCMPDEF                      ; Define DAP Access Complete message
                   0000             $DAPDATDEF                      ; Define DAP Data message
                   0000             $DAPSTSDEF                      ; Define DAP Status message
                   0000             $DAPKEYDEF                      ; Define DAP Key Definition message
                   0000             $DAPALLDEF                      ; Define DAP Allocation message
                   0000             $DAPSUMDEF                      ; Define DAP Summary message
                   0000             $DAPTIMDEF                      ; Define DAP Date and Time message
                   0000             $DAPPRODEF                      ; Define DAP Protection message
                   0000             $DAPNAMDEF                      ; Define DAP Name message
                   0000             $DAPFIDDEF                      ; Define DAP field ID symbols
                   0000
                   0000    ;
                   0000    ; Macros:
                   0000    ;
                   0000    ;        See next page for local macro definitions.
                   0000    ;
                   0000    ; Equated Symbols:
                   0000    ;
                   0000
         00000000  0000    K_EXT=0                                  ; Extensible field format
         00000001  0000    K_FIX=1                                  ; Fixed length field format
         00000002  0000    K_IMG=2                                  ; Image field format
         00000003  0000    K_ROM=3                                  ; Rest-of-message field format
                   0000
         00000004  0000    V_DESC=4                                 ; Store descriptor of source field
         00000005  0000    V_TRUNC=5                                ; Truncate source field if necessary
         00000006  0000    V_SRCR3=6                                ; Source field size in R3
                   0000                                             ;  (applicable only if K_FIX specified)
                   0000
         00000010  0000    M_DESC=<1@V_DESC>                        ; Mask for V_DESC
         00000020  0000    M_TRUNC=<1@V_TRUNC>                      ; Mask for V_TRUNC
         00000040  0000    M_SRCR3=<1@V_SRCR3>                      ; Mask for V_SRCR3
                   0000
                   0000             ASSUME  DAP$Q_DCODE_FLG EQ 0
                   0000
                   0000    ;
                   0000    ; Own Storage:
                   0000    ;
                   0000    ;        None
                   0000    ;
```

```
0000            .SBTTL  LOCAL MACRO DEFINITIONS
0000
0000    ;++
0000    ; STORE_FIELD obtains the next field (if any) from the DAP message being parsed,
0000    ; converts it to an appropriate format, and stores the result in the designated
0000    ; field of the DAP control block. The arguments (coded in-line) are:
0000    ;
0000    ;       NAME  = the symbolic name of DAP field used to generate symbolic DAP
0000    ;               control block offset and field ID values.
0000    ;       SIZE  = the size in bytes of designated field in DAP control block.
0000    ;       FORMAT= the format or structure of the source field. Choices are:
0000    ;               K_EXT = extensible field (bit7 of each byte is used to signify
0000    ;                       termination/continuation (0/1) of the field).
0000    ;               K_FIX = fixed length field.
0000    ;               K_IMG = image field (counted string).
0000    ;               K_ROM = rest-of-message is taken as the next field.
0000    ;       MASK  = the flags to control field processing:
0000    ;               M_DESC= store only descriptor of the source field.
0000    ;               M_TRUNC=truncate extra bytes if SRC field size is larger than
0000    ;                       DST field size (instead of declaring an error).
0000    ;               M_SRCR3=size of source field is given in R3 (applicable only if
0000    ;                       K_FIX is also specified).
0000    ;--
0000
0000            .MACRO  STORE_FIELD NAME,SIZE=1,FORMAT=1,MASK=0
0000            BSBW    STORE_FIELD
0000            .BYTE   SIZE
0000    TMP1..=.
0000            .IIF EQ <SIZE-1>,       .BYTE   DAP$B_'NAME
0000            .IIF EQ <SIZE-2>,       .BYTE   DAP$W_'NAME
0000            .IIF EQ <SIZE-4>,       .BYTE   DAP$L_'NAME
0000            .IIF EQ <SIZE-6>,       .BYTE   DAP$W_'NAME
0000            .IIF EQ <SIZE-8>,       .BYTE   DAP$Q_'NAME
0000    TMP2..=.
0000            .IIF EQ <TMP2..-TMP1..>,.ERROR  ;***** invalid field size *****;
0000            .BYTE   DAP$_'NAME
0000            .BYTE   FORMAT!MASK
0000            .ENDM   STORE_FIELD
0000
0000    ;++
0000    ; CHECK_MASKS examines the designated field of the DAP control block for
0000    ; invalid and unsupported bits set. The arguments (coded in-line) are:
0000    ;
0000    ;       NAME  = the symbolic name of the DAP field used to generate symbolic
0000    ;               invalid and unsupported mask values.
0000    ;       SIZE  = the size in bytes of designated field in the DAP control block.
0000    ;--
0000
0000            .MACRO  CHECK_MASKS NAME,SIZE=1
0000            BSBW    CHECK_MASKS
0000            .BYTE   SIZE
0000    TMP1..=.
0000            .IIF EQ <SIZE-1>,       .BYTE   DAP$K_'NAME'_I,DAP$K_'NAME'_U
0000            .IIF EQ <SIZE-2>,       .WORD   DAP$K_'NAME'_I,DAP$K_'NAME'_U
0000            .IIF EQ <SIZE-4>,       .LONG   DAP$K_'NAME'_I,DAP$K_'NAME'_U
0000    TMP2..=.
0000            .IIF EQ <TMP2..-TMP1..>,.ERROR  ;***** invalid field size *****;
```

```
                    0000   186          .ENDM   CHECK_MASKS
```

```
           0000          .SBTTL  FAL$DECODE_MSG - DECODE DAP MESSAGE
       00000000          .PSECT  FAL$CODE        NOSHR,EXE,RD,NOWRT,BYTE
           0000
           0000  ;++
           0000  ; Functional Description:
           0000  ;
           0000  ;       FAL$DECODE_MSG is responsible for parsing a DAP message into its
           0000  ;       constituent fields, storing these field values into corresponding fields
           0000  ;       in the DAP control block, and finally performing validity checks on the
           0000  ;       contents of the converted fields to screen out invalid and unsupported
           0000  ;       bit options or field values.
           0000  ;
           0000  ;       Each DAP message logically consists of two parts:
           0000  ;       (1) a message header (called the operator field in DAP).
           0000  ;       (2) a message body (called the operand field in DAP).
           0000  ;       In addition, the message header may optionally contain a system
           0000  ;       specific field for use by homogeneous systems which is treated as a
           0000  ;       mini-message with discrete fields.
           0000  ;
           0000  ; Calling Sequence:
           0000  ;
           0000  ;       CALLS   #1,FAL$DECODE_MSG
           0000  ;
           0000  ; Input Parameters:
           0000  ;
           0000  ;       4(AP)   Address of DAP control block
           0000  ;
           0000  ; Implicit Inputs:
           0000  ;
           0000  ;       None
           0000  ;
           0000  ; Output Parameters:
           0000  ;
           0000  ;       R0      Status code
           0000  ;       R1      Destroyed
           0000  ;
           0000  ; Implicit Outputs:
           0000  ;
           0000  ;       Various fields of the DAP control block are updated.
           0000  ;
           0000  ; Completion Codes:
           0000  ;
           0000  ;       DAP$L_DCODE_STS is returned in R0 where bit 0 indicates success/failure.
           0000  ;
           0000  ; Side Effects:
           0000  ;
           0000  ;       None
           0000  ;
           0000  ;--
           0000
   0FFC    0000          .ENTRY  FAL$DECODE_MSG,^M<R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
           0002                                          ; Entry point
           0002
           0002  ;
           0002  ; Perform initialization.
           0002  ;
           0002
```

```
59    04 AC    D0  0002          MOVL    4(AP),R9                 ; Get address of DAP control block
18 A9    01    D0  0006          MOVL    #1,DAP$L_DCODE_STS(R9)   ; Assume successful parse
5A    08 A9    7D  000A          MOVQ    DAP$Q_MSG_BUF1(R9),R10   ; R10 = size of message
                   000E                                           ; R11 = address of start-of-message
10 A9    5A    7D  000E          MOVQ    R10,DAP$Q_MSG_BUF2(R9)   ; Store in result descriptor
   5A    5B    C0  0012          ADDL2   R11,R10                  ; R10 = address of end-of-message + 1
                   0015          $ZERO_FILL-                      ; Zero current message work area
                   0015                  DST=DAP$L_CMWA(R9)-      ;  in DAP control block
                   0015                  SIZE=#DAP$K_CMWA         ;
      24 A9    94  001F          CLRB    DAP$B_X_FIELD(R9)        ; Clear 'explicitly specified' flags
                   0022
                   0022 ;+
                   0022 ; Note the current status of registers R8-R11:
                   0022 ;
                   0022 ;        R8      Currently undefined; later it will be used to contain the
                   0022 ;                address of the routine to execute on reaching end-of-message
                   0022 ;        R9      Address of DAP control block
                   0022 ;        R10     Address of end-of-message-buffer + 1; later it will contain
                   0022 ;                the address of end-of-message + 1
                   0022 ;        R11     Address of start-of-message; it will be continually updated
                   0022 ;                to contain the address of the next byte to parse
                   0022 ;-
```

```
                        0022            .SBTTL  HEADER - DECODE MESSAGE HEADER
                        0022
                        0022    ;++
                        0022    ; Decode the header of the DAP message (operator portion of the message).
                        0022    ; Then dispatch on message type to parse the body of the DAP message (operand
                        0022    ; portion of the message).
                        0022    ;--
                        0022
                        0022    HEADER:                                         ; Continuation of mainline
                        0022
                        0022            ASSUME  DAP$K_CNF_MSG EQ 1
                        0022            ASSUME  DAP$K_ATT_MSG EQ 2
                        0022            ASSUME  DAP$K_ACC_MSG EQ 3
                        0022            ASSUME  DAP$K_CTL_MSG EQ 4
                        0022            ASSUME  DAP$K_CON_MSG EQ 5
                        0022            ASSUME  DAP$K_ACK_MSG EQ 6
                        0022            ASSUME  DAP$K_CMP_MSG EQ 7
                        0022            ASSUME  DAP$K_DAT_MSG EQ 8
                        0022            ASSUME  DAP$K_STS_MSG EQ 9
                        0022            ASSUME  DAP$K_KEY_MSG EQ 10
                        0022            ASSUME  DAP$K_ALL_MSG EQ 11
                        0022            ASSUME  DAP$K_SUM_MSG EQ 12
                        0022            ASSUME  DAP$K_TIM_MSG EQ 13
                        0022            ASSUME  DAP$K_PRO_MSG EQ 14
                        0022            ASSUME  DAP$K_NAM_MSG EQ 15
                        0022
                        0022    ;
                        0022    ; For optional fields, apply default values as appropriate.
                        0022    ;
                        0022
   3C A9    69      7E  0022            MOVAQ   (R9),DAP$Q_SYSPEC+4(R9) ; Initialize descriptor
                        0026
                        0026    ;
                        0026    ; Process the DAP message type field (required).
                        0026    ;
                        0026
 58    091D'CF      9E  0026            MOVAB   W^ERROR_FORMAT,R8       ; Specify transfer address on EOM
                        002B            STORE_FIELD     TYPE,1,K_FIX    ; Save type field
            66      95  0032            TSTB    (R6)                    ; Test for valid value
            0B      13  0034            BEQL    10$                     ; Branch if out-of-range
       0F   66      91  0036            CMPB    (R6),#DAP$K_NAM_MSG     ; Test for valid value
            06      1A  0039            BGTRU   10$                     ; Branch if out-of-range
   1A A9    66      90  003B            MOVB    (R6),DAP$B_DCODE_MSG(R9); Return message type in status code
            03      11  003F            BRB     20$                     ; Continue
          08E8      31  0041    10$:    BRW     ERROR_INVALID           ; Branch aid
                        0044
                        0044    ;+
                        0044    ; Process the DAP message flags field (required for most messages).
                        0044    ; This is a combination menu and bit option field whereby each bit set denotes
                        0044    ; that either an associated field is included in the message or a message
                        0044    ; option is specified.
                        0044    ;
                        0044    ; Note: If no flags field is found (i.e., its a one-byte message), the
                        0044    ;       associated operand parse routine for the message will still be entered
                        0044    ;       (via DISPATCH_TABLE) to determine if the message is valid and to apply
                        0044    ;       operand field default values.
                        0044    ;-
                        0044
                        0044         ASSUME  DAP$V_STREAMID+1 EQ DAP$V_LENGTH
                        0044         ASSUME  DAP$V_LENGTH+1 EQ DAP$V_LEN256
                        0044         ASSUME  DAP$V_LEN256+1 EQ DAP$V_BITCNT
                        0044         ASSUME  DAP$V_BITCNT+2 EQ DAP$V_SYSPEC
                        0044         ASSUME  DAP$V_SYSPEC+1 EQ DAP$V_SEGMENT
                        0044
      58  00C8'CF   9E  0044  20$:   MOVAB   W^DISPATCH_TABLE,R8          ; Specify transfer address on EOM
                        0049         STORE_FIELD     FLAGS,1,K_EXT        ; Save flags field
                        0050         CHECK_MASKS     FLAGS,1              ; Validate bit options
      58  091D'CF   9E  0056         MOVAB   W^ERROR_FORMAT,R8            ; Specify transfer address on EOM
              5C  66  9A  005B       MOVZBL  (R6),AP                      ; Copy menu to scratch register
                        005E  HDR_LOOP:
50    5C  07  00    EA  005E         FFS     #0,#DAP$V_SEGMENT+1,AP,R0; Get position of next bit set
                        0063         $CLRBIT R0,AP                        ; Clear menu bit just found
             F4 AF  9F  0067         PUSHAB  B^HDR_LOOP                   ; Push return address on stack
                        006A         $CASEB  SELECTOR=R0-                 ; Next field/option:
                        006A                 DISPL=<-                     ;
                        006A                     10$-                     ;  STREAMID
                        006A                     20$-                     ;  LENGTH
                        006A                     30$-                     ;  LEN256
                        006A                     ERROR_UNSUPPORT-         ;  BITCNT
                        006A                     ERROR_FORMAT-            ;  Reserved
                        006A                     60$-                     ;  SYSPEC
                        006A                     ERROR_UNSUPPORT-         ;  SEGMENT
                        006A                 >
                4A  11  007C         BRB     DISPATCH_TABLE               ; Message header syntax is correct
                        007E
                        007E  ;
                        007E  ; Process each field/option specified in the menu (optional).
                        007E  ;
                        007E
                        007E  10$:   STORE_FIELD     STREAMID,1,K_FIX; Save data stream identification field
                66  95  0085         TSTB    (R6)                         ; Currently, multi-streams are
                        0087                                              ;  not supported, so check value
                3C  12  0087         BNEQ    HDR_UNSUPPORT                ; Branch on error
                    05  0089         RSB
                        008A  20$:   STORE_FIELD     LENGTH,1,K_FIX       ; Save length field
         08 5C  02  E1  0091         BBC     #DAP$V_LEN256,AP,35$         ; Branch if length value in header is
                        0095                                              ;  expressed in one byte (i.e., there
                    05  0095         RSB                                  ;  is no LEN256 field present)
                        0096  30$:   STORE_FIELD     LEN256,1,K_FIX       ; Save length extension field
                        009D
                        009D  ;
                        009D  ; Determine end-of-message based on operand length value in message header.
                        009D  ;
                        009D
                        009D         ASSUME  DAP$B_LENGTH+1 EQ DAP$B_LEN256
                        009D
    50     33 A9    3C  009D  35$:   MOVZWL  DAP$B_LENGTH(R9),R0          ; Get operand length value
51     5B  50    C1  00A1           ADDL3   R0,R11,R1                    ; Compute new end-of-message + 1 address
       5A  51    D1  00A5           CMPL    R1,R10                       ; Error if not enough bytes in buffer
           18    1A  00A8           BGTRU   HDR_INVALID                  ;  to contain message
       5A  51    D0  00AA           MOVL    R1,R10                       ; Update end-of-message address
                 05  00AD           RSB                                  ;
                     00AE
                     00AE  ;+
                          00AE  ; Suggested code to support the BITCNT field is shown below.
                          00AE  ;
                          00AE  ;40$:     STORE_FIELD     BITCNT,1,K_FIX   ; Save bit count field
                          00AE  ;         CMPB    DAP$B_TYPE(R9),-         ; BITCNT field allowed only in
                          00AE  ;                 #DAP$K_DAT_MSG           ;  Data message
                          00AE  ;         BNEQ    80$                      ; Branch on error
                          00AE  ;         CMPB    (R6),#7                  ; Check for value in the range 0-7
                          00AE  ;         BGTRU   HDR_INVALID              ; Branch on error
                          00AE  ;         RSB                              ;
                          00AE  ;-
                          00AE
                          00AE  60$:      STORE_FIELD     SYSPEC,8,K_IMG,<M_DESC>
                          00B5                                             ; Save descriptor of system specific
                          00B5                                             ;  field
            30 A9   91    00B5            CMPB    DAP$B_TYPE(R9),-         ; SYSPEC field not allowed in
               01         00B8                    #DAP$K_CNF_MSG           ;  Configuration message
               05   13    00B9            BEQL    80$                      ; Branch on error
         01 69 34   E1    00BB            BBC     #DAP$V_VAXVMS,(R9),80$   ; SYSPEC field allowed only if
                          00BF                                             ;  systems are homogeneous
                    05    00BF            RSB                              ;
               68   17    00C0  80$:      JMP     (R8)                     ; Branch to error_format routine
                          00C2
                          00C2  ;
                          00C2  ; Branch here on exception condition.
                          00C2  ;
                          00C2
                          00C2  HDR_INVALID:                               ;
             0867   31    00C2            BRW     ERROR_INVALID            ; Branch aid
                          00C5  HDR_UNSUPPORT:                             ;
             0870   31    00C5            BRW     ERROR_UNSUPPORT          ; Branch aid
```

```
               00C8          .SBTTL  DISPATCH_TABLE - CASE ON MESSAGE TYPE
               00C8
               00C8   ;+
               00C8   ; The DAP messasge header has been successfully parsed. Now dispatch on message
               00C8   ; type to the appropriate code segment to process the body of the message.
               00C8   ;
               00C8   ; Note: The case table entries below should match the DAP$K_VALID_R2F message
               00C8   ;       mask!
               00C8   ;-
               00C8
               00C8   DISPATCH_TABLE:                                ; Continuation of mainline
57   00   9A   00C8           MOVZBL  #DAP$_UNKNOWN,R7               ; Set field ID to 'unknown'
               00CB           $CASEB  SELECTOR=DAP$B_TYPE(R9)-       ; Dispatch to message specific decode
               00CB                   BASE=#DAP$K_CNF_MSG-           ;  routine to process:
               00CB                   DISPL=<-                       ;
               00CB                           CNF_MSG-               ;  Configuration message
               00CB                           ATT_MSG-               ;  Attributes message
               00CB                           ACC_MSG-               ;  Access message
               00CB                           CTL_MSG-               ;  Control message
               00CB                           CON_MSG-               ;  Continue Transfer message
               00CB                           ERROR_SYNC-            ;  Acknowledge message
               00CB                           CMP_MSG-               ;  Access Complete message
               00CB                           DAT_MSG-               ;  Data message
               00CB                           ERROR_SYNC-            ;  Status message
               00CB                           KEY_MSG-               ;  Key Definition message
               00CB                           ALL_MSG-               ;  Allocation message
               00CB                           ERROR_SYNC-            ;  Summary message
               00CB                           TIM_MSG-               ;  Date and Time message
               00CB                           PRO_MSG-               ;  Protection message
               00CB                           NAM_MSG-               ;  Name message
               00CB                   >                              ;
               00EE
               00EE   ;
               00EE   ; The message type value has been validated (bounds checked), so the type value
               00EE   ; will not be outside the range of the case table above.
               00EE   ;
```

```
                        00EE             .SBTTL  CNF_MSG - DECODE CONFIGURATION MESSAGE
                        00EE
                        00EE    ;++
                        00EE    ; Decode the operand fields of the Configuration message.
                        00EE    ;--
                        00EE
                        00EE    CNF_MSG:                                       ; Code segment of mainline
        58  091D'CF  9E 00EE             MOVAB   W^ERROR_FORMAT,R8             ; Specify transfer address on EOM
                        00F3
                        00F3    ;
                        00F3    ; Process the buffer size field (required).
                        00F3    ;
                        00F3
                        00F3             STORE_FIELD     BUFSIZ,2,K_FIX  ; Save buffer size field
                        00FA
                        00FA    ;
                        00FA    ; Process system software and DAP protocol version number fields (required).
                        00FA    ; These fields are for information purposes only; hence no bounds checking
                        00FA    ; on their values is performed.
                        00FA    ;
                        00FA
                        00FA             STORE_FIELD     OSTYPE,1,K_FIX  ; Save operating system type field
                        0101             STORE_FIELD     FILESYS,1,K_FIX ; Save file system type field
                        0108             STORE_FIELD     VERNUM,1,K_FIX  ; Save DAP version # field
                        010F             STORE_FIELD     ECONUM,1,K_FIX  ; Save ECO version # field
                        0116             STORE_FIELD     USRNUM,1,K_FIX  ; Save user protocol version # field
                        011D             STORE_FIELD     DECVER,1,K_FIX  ; Save DEC software version # field
                        0124             STORE_FIELD     USRVER,1,K_FIX  ; Save user software version # field
                        012B
                        012B    ;
                        012B    ; Process the system capabilities field (required).
                        012B    ; Bits set that are not defined in the DAP spec are ignored (not flagged as
                        012B    ; an error) to facilitate compatibility with earlier implementations of DAP.
                        012B    ;
                        012B
                        012B             STORE_FIELD     SYSCAP,8,K_EXT,<M_TRUNC>
                        0132                                                   ; Save system capabilities field
                        0132
                        0132    CHECK_PROTOCOL_VERSION:                        ; Set appropriate DAP$Q_DCODE_FLG bits
          50   44 A9  9A 0132             MOVZBL  DAP$B_VERNUM(R9),R0           ; Combine version number and ECO
       50   50   08  78 0136             ASHL    #8,R0,R0                      ;  number fields into one 16-bit
          50   45 A9  80 013A             ADDB2   DAP$B_ECONUM(R9),R0           ;  value for easy comparison
                        013E
                        013E    ;
                        013E    ; Set status flag if partner implemented to DAP spec since V4.1.
                        013E    ;
                        013E
     0401 8F     50  B1 013E             CMPW    R0,#^X0401                    ; Did partner implement since DAP V4.1?
                 51  1F 0143             BLSSU   10$                           ; Branch if not
                        0145             $SETBIT #DAP$V_GEQ_V41,(R9)           ; Set flag
                        0149
                        0149    ;
                        0149    ; Set status flag if partner implemented to DAP spec since V4.2.
                        0149    ;
                        0149
     0402 8F     50  B1 0149             CMPW    R0,#^X0402                    ; Did partner implement since DAP V4.2?
                 46  1F 014E             BLSSU   10$                           ; Branch if not
```

```
                       0150          $SETBIT #DAP$V_GEQ_V42,(R9)      ; Set flag
                       0154
                       0154     ;
                       0154     ; Set status flag if partner implemented to DAP spec since V5.2.
                       0154     ;
                       0154
0502 8F   50    B1     0154          CMPW    R0,#^X0502               ; Did partner implement since DAP V5.2?
          3B    1F     0159          BLSSU   10$                      ; Branch if not
                       015B          $SETBIT #DAP$V_GEQ_V52,(R9)      ; Set flag
                       015F
                       015F     ;
                       015F     ; Set status flag if partner implemented to DAP spec since V5.4.
                       015F     ;
                       015F
0504 8F   50    B1     015F          CMPW    R0,#^X0504               ; Did partner implement since DAP V5.4?
          30    1F     0164          BLSSU   10$                      ; Branch if not
                       0166          $SETBIT #DAP$V_GEQ_V54,(R9)      ; Set flag
                       016A
                       016A     ;
                       016A     ; Set status flag if partner implemented to DAP spec since V5.6.
                       016A     ;
                       016A
0506 8F   50    B1     016A          CMPW    R0,#^X0506               ; Did partner implement since DAP V5.6?
          25    1F     016F          BLSSU   10$                      ; Branch if not
                       0171          $SETBIT #DAP$V_GEQ_V56,(R9)      ; Set flag
                       0175
                       0175     ;
                       0175     ; Set status flag if partner implemented to DAP spec since V6.0.
                       0175     ;
                       0175
0600 8F   50    B1     0175          CMPW    R0,#^X0600               ; Did partner implement since DAP V6.0?
          1A    1F     017A          BLSSU   10$                      ; Branch if not
                       017C          $SETBIT #DAP$V_GEQ_V60,(R9)      ; Set flag
                       0180
                       0180     ;
                       0180     ; Set status flag if partner implemented to DAP spec since V7.0.
                       0180     ;
                       0180
0700 8F   50    B1     0180          CMPW    R0,#^X0700               ; Did partner implement since DAP V7.0?
          0F    1F     0185          BLSSU   10$                      ; Branch if not
                       0187          $SETBIT #DAP$V_GEQ_V70,(R9)      ; Set flag
                       018B
                       018B     ;
                       018B     ; Set status flag if partner implemented to DAP spec since V7.1.
                       018B     ;
                       018B
0701 8F   50    B1     018B          CMPW    R0,#^X0701               ; Did partner implement since DAP V7.1?
          04    1F     0190          BLSSU   10$                      ; Branch if not
                       0192          $SETBIT #DAP$V_GEQ_V71,(R9)      ; Set flag
                       0196
                       0196     ;
                       0196     ; Set experimental protocol flags from the low order four bits of the USRNUM
                       0196     ; field only if partner is VAX/VMS.
                       0196     ;
                       0196
       42 A9    91     0196     10$:     CMPB    DAP$B_OSTYPE(R9),-       ; Branch if partner is not VAX/VMS
          07           0199                  #DAP$K_VAXVMS            ;
```

```
                    0B   12  019A          BNEQ     CHECK_FILE_SYSTEM             ;
50   46 A9   04     00   EF  019C          EXTZV    #0,#4,DAP$B_USRNUM(R9),R0    ; Get low order four bits of USRNUM
   69   04   2C     50   F0  01A2          INSV     R0,#DAP$V_VMS_XPF1,#4,(R9)   ; Set experimental protocol flags
                             01A7
                             01A7  CHECK_FILE_SYSTEM:                            ; Set appropriate DAP$Q_DCODE_FLG bit.
                             01A7
                             01A7          ASSUME   DAP$K_RMS11 EQ 1
                             01A7          ASSUME   DAP$K_RMS20 EQ 2
                             01A7          ASSUME   DAP$K_RMS32 EQ 3
                             01A7          ASSUME   DAP$K_FCS11 EQ 4
                             01A7          ASSUME   DAP$K_RT11FS EQ 5
                             01A7          ASSUME   DAP$K_NO_FS EQ 6
                             01A7          ASSUME   DAP$K_TOPS20FS EQ 7
                             01A7          ASSUME   DAP$K_TOPS10FS EQ 8
                             01A7          ASSUME   DAP$K_RMS32S EQ 10
                             01A7
                             01A7  ;
                             01A7  ; Set status flag pertaining to the type of file system used by the remote node.
                             01A7  ;
                             01A7
                             01A7          $CASEB   SELECTOR=DAP$B_FILESYS(R9)-
                             01A7                   BASE=#DAP$K_RMS11-           ; Type of remote file system:
                             01A7                   DISPL=<-                     ;
                             01A7                       10$-                     ;  RMS-11
                             01A7                       10$-                     ;  RMS-20
                             01A7                       10$-                     ;  RMS-32
                             01A7                       20$-                     ;  FCS-11
                             01A7                       30$-                     ;  RT-11
                             01A7                       40$-                     ;  No file system present
                             01A7                       30$-                     ;  TOPS-20
                             01A7                       30$-                     ;  TOPS-10
                             01A7                       40$-                     ;  Undefined
                             01A7                       10$-                     ;  RMS-32 subset
                             01A7                   >                            ;
                    10   11  01C0          BRB      40$                          ;
                             01C2  10$:    $SETBIT  #DAP$V_RMS,(R9)              ; Set RMS based file system flag
                    0A   11  01C6          BRB      40$                          ;
                             01C8  20$:    $SETBIT  #DAP$V_FCS,(R9)              ; Set FCS based file system flag
                    04   11  01CC          BRB      40$                          ;
                             01CE  30$:    $SETBIT  #DAP$V_STM_ONLY,(R9)         ; Set stream ASCII file system flag
                             01D2  40$:                                          ;
                             01D2
                             01D2  CHECK_OPERATING_SYSTEM:                       ; Set appropriate DAP$Q_DCODE_FLG bit
                             01D2
                             01D2          ASSUME   DAP$K_RT11 EQ 1
                             01D2          ASSUME   DAP$K_RSTS EQ 2
                             01D2          ASSUME   DAP$K_RSX11S EQ 3
                             01D2          ASSUME   DAP$K_RSX11M EQ 4
                             01D2          ASSUME   DAP$K_RSX11D EQ 5
                             01D2          ASSUME   DAP$K_IAS EQ 6
                             01D2          ASSUME   DAP$K_VAXVMS EQ 7
                             01D2          ASSUME   DAP$K_TOPS20 EQ 8
                             01D2          ASSUME   DAP$K_TOPS10 EQ 9
                             01D2          ASSUME   DAP$K_RSX11MP EQ 12
                             01D2          ASSUME   DAP$K_COPOS11 EQ 13
                             01D2          ASSUME   DAP$K_P_OS EQ 14
                             01D2          ASSUME   DAP$K_VAXELAN EQ 15
```

```
                01D2
                01D2      ;
                01D2      ; Set status flag pertaining to the type of operating system being used at the
                01D2      ; remote node.
                01D2      ;
                01D2
                01D2              $CASEB  SELECTOR=DAP$B_OSTYPE(R9)-
                01D2                      BASE=#DAP$K_RT11-          ; Type of remote operating system:
                01D2                      DISPL=<-
                01D2                              50$-               ; RT-11
                01D2                              60$-               ; RSTS/E
                01D2                              70$-               ; RSX-11S
                01D2                              70$-               ; RSX-11M
                01D2                              80$-               ; RSX-11D (classified as IAS)
                01D2                              80$-               ; IAS
                01D2                              10$-               ; VAX/VMS
                01D2                              40$-               ; TOPS-20
                01D2                              30$-               ; TOPS-10
                01D2                              100$-              ; Undefined
                01D2                              100$-              ; Undefined
                01D2                              70$-               ; RSX-11M-PLUS
                01D2                              40$-               ; TOPS-20 (using 2050/2060 front end)
                01D2                              90$-               ; P/OS
                01D2                              20$-               ; VAXELAN
                01D2                      >                          ;
  34     11     01F5              BRB     100$                       ;
                01F7      10$:    $SETBIT #DAP$V_VAXVMS,(R9)         ; Set VAX/VMS system flag
  2E     11     01FB              BRB     100$                       ;
                01FD      20$:    $SETBIT #DAP$V_VAXELAN,(R9)        ; Set VAXELAN system flag
  28     11     0201              BRB     100$                       ;
                0203      30$:    $SETBIT #DAP$V_TOPS10,(R9)         ; Set TOPS-10 system flag
  22     11     0207              BRB     100$                       ;
                0209      40$:    $SETBIT #DAP$V_TOPS20,(R9)         ; Set TOPS-20 and COPOS11 system flag
  1C     11     020D              BRB     100$                       ;
                020F      50$:    $SETBIT #DAP$V_RT11,(R9)           ; Set RT-11 system flag
  16     11     0213              BRB     100$                       ;
                0215      60$:    $SETBIT #DAP$V_RSTS,(R9)           ; Set RSTS/E system flag
  10     11     0219              BRB     100$                       ;
                021B      70$:    $SETBIT #DAP$V_RSX,(R9)            ; Set RSX-11S, RSX-11M, and RSX-11M-PLUS
  0A     11     021F              BRB     100$                       ;  system flag
                0221      80$:    $SETBIT #DAP$V_IAS,(R9)            ; Set IAS and RSX-11D system flag
  04     11     0225              BRB     100$                       ;
                0227      90$:    $SETBIT #DAP$V_P_OS,(R9)           ; Set PO/S system flag
                022B
0717     31     022B      100$:   BRW     EXIT_SUCCESS               ; Message syntax is correct
```

```
                           022E          .SBTTL  ATT_MSG - DECODE ATTRIBUTES MESSAGE
                           022E
                           022E  ;++
                           022E  ; Decode the operand fields of the Attributes message.
                           022E  ;--
                           022E
                           022E  ATT_MSG:                                    ; Code segment of mainline
                           022E
                           022E  ;
                           022E  ; For optional fields, apply default values as appropriate.
                           022E  ;
                           022E
       44 A9   02    90    022E          MOVB    #DAP$K_DATATYP_D,DAP$B_DATATYPE(R9)
       45 A9   00    90    0232          MOVB    #DAP$K_ORG_D,DAP$B_ORG(R9)
       46 A9   01    90    0236          MOVB    #DAP$K_RFM_D,DAP$B_RFM(R9)
  48 A9    0200 8F   B0    023A          MOVW    #DAP$K_BLS_D,DAP$W_BLS(R9)
       52 A9   08    90    0240          MOVB    #DAP$K_BSZ_D,DAP$B_BSZ(R9)
       60 A9   69    7E    0244          MOVAQ   (R9),DAP$Q_RUNSYS+4(R9) ; Initialize descriptor
                           0248
                           0248  ;
                           0248  ; Process the attributes menu field (optional).
                           0248  ; Each bit set denotes that its associated field follows in the message.
                           0248  ;
                           0248
                           0248          ASSUME  DAP$V_DATATYPE+1 EQ DAP$V_ORG
                           0248          ASSUME  DAP$V_ORG+1 EQ DAP$V_RFM
                           0248          ASSUME  DAP$V_RFM+1 EQ DAP$V_RAT
                           0248          ASSUME  DAP$V_RAT+1 EQ DAP$V_BLS
                           0248          ASSUME  DAP$V_BLS+1 EQ DAP$V_MRS
                           0248          ASSUME  DAP$V_MRS+1 EQ DAP$V_ALQ1
                           0248          ASSUME  DAP$V_ALQ1+1 EQ DAP$V_BKS
                           0248          ASSUME  DAP$V_BKS+1 EQ DAP$V_FSZ
                           0248          ASSUME  DAP$V_FSZ+1 EQ DAP$V_MRN
                           0248          ASSUME  DAP$V_MRN+1 EQ DAP$V_RUNSYS
                           0248          ASSUME  DAP$V_RUNSYS+1 EQ DAP$V_DEQ1
                           0248          ASSUME  DAP$V_DEQ1+1 EQ DAP$V_FOP1
                           0248          ASSUME  DAP$V_FOP1+1 EQ DAP$V_BSZ
                           0248          ASSUME  DAP$V_BSZ+1 EQ DAP$V_DEV
                           0248          ASSUME  DAP$V_DEV+2 EQ DAP$V_LRL
                           0248          ASSUME  DAP$V_LRL+1 EQ DAP$V_HBK
                           0248          ASSUME  DAP$V_HBK+1 EQ DAP$V_EBK
                           0248          ASSUME  DAP$V_EBK+1 EQ DAP$V_FFB
                           0248          ASSUME  DAP$V_FFB+1 EQ DAP$V_SBN
                           0248
        58   0945'CF   9E  0248          MOVAB   W^EXIT_SUCCESS,R8           ; All done if end-of-message
                           024D          STORE_FIELD     ATTMENU,4,K_EXT     ; Save attributes menu field
                           0254          CHECK_MASKS     ATTMENU,4           ; Validate bit options
        58   091D'CF   9E  0260          MOVAB   W^ERROR_FORMAT,R8           ; Specify transfer address on EOM
             5C   66   D0  0265          MOVL    (R6),AP                     ; Copy menu to scratch register
                           0268  ATT_LOOP:                                   ;
   50   5C   15   00   EA  0268          FFS     #0,#DAP$V_SBN+1,AP,R0       ; Get position of next bit set
                           026D          $CLRBIT R0,AP                       ; Clear menu bit just found
             F4 AF     9F  0271          PUSHAB  B^ATT_LOOP                  ; Push return address on stack
                           0274          $CASEB  SELECTOR=R0-                ; Next field:
                           0274                  DISPL=<-                    ;
                           0274                          10$-                ;  DATATYPE
                           0274                          20$-                ;  ORG
```

```
                0274                            30$-                    ; RFM
                0274                            40$-                    ; RAT
                0274                            50$-                    ; BLS
                0274                            60$-                    ; MRS
                0274                            70$-                    ; ALQ1
                0274                            80$-                    ; BKS
                0274                            90$-                    ; FSZ
                0274                            100$-                   ; MRN
                0274                            110$-                   ; RUNSYS
                0274                            120$-                   ; DEQ1
                0274                            130$-                   ; FOP1
                0274                            140$-                   ; BSZ
                0274                            150$-                   ; DEV
                0274                            ERROR_FORMAT-           ; Reserved
                0274                            170$-                   ; LRL
                0274                            180$-                   ; HBK
                0274                            190$-                   ; EBK
                0274                            200$-                   ; FFB
                0274                            210$-                   ; SBN
                0274                        >
     06A0  31   02A2          BRW      EXIT_SUCCESS                 ; Message syntax is correct
                02A5
                02A5    ;
                02A5    ; Process each field specified in the menu (optional).
                02A5    ;
                02A5
                02A5    40$:  STORE_FIELD     RAT,1,K_EXT         ; Save record attributes field
                02AC          CHECK_MASKS     RAT,1               ; Validate bit options
           05   02B2          RSB                                 ;
                02B3    50$:  STORE_FIELD     BLS,2,K_FIX         ; Save block size field
           05   02BA          RSB                                 ;
                02BB    60$:  STORE_FIELD     MRS,2,K_FIX         ; Save maximum record size field
           05   02C2          RSB                                 ;
                02C3    70$:  STORE_FIELD     ALQ1,4,K_IMG        ; Save allocation quantity field
           05   02CA          RSB                                 ;
                02CB    80$:  STORE_FIELD     BKS,1,K_FIX         ; Save bucket size field
           05   02D2          RSB                                 ;
                02D3    90$:  STORE_FIELD     FSZ,1,K_FIX         ; Save fixed control area size field
           05   02DA          RSB                                 ;
                02DB    100$: STORE_FIELD     MRN,4,K_IMG         ; Save maximum record number field
           05   02E2          RSB                                 ;
                02E3    110$: STORE_FIELD     RUNSYS,8,K_IMG,<M_DESC>
                02EA                                              ; Save descriptor of run-time
                02EA                                              ;  system string
           05   02EA          RSB                                 ;
                02EB    120$: STORE_FIELD     DEQ1,2,K_FIX        ; Save default extension quantity field
           05   02F2          RSB                                 ;
                02F3    130$: STORE_FIELD     FOP1,4,K_EXT        ; Save file options field
                02FA          CHECK_MASKS     FOP,4               ; Validate bit options
           05   0306          RSB                                 ;
                0307    10$:  STORE_FIELD     DATATYPE,1,K_EXT;   Save data type field
                030E          CHECK_MASKS     DATATYP,1           ; Validate bit options
           05   0314          RSB                                 ;
                0315    20$:  STORE_FIELD     ORG,1,K_FIX         ; Save file organization field
                031C
                031C          ASSUME  DAP$K_SEQ EQ 0
                031C          ASSUME  DAP$K_REL EQ 16
```

```
                        031C   781            ASSUME  DAP$K_IDX EQ 32
                        031C   782
              66    95  031C   783            TSTB    (R6)                    ; Check for valid value
              0C    13  031E   784            BEQL    25$                     ; Branch if ok
        10    66    91  0320   785            CMPB    (R6),#DAP$K_REL         ; Check for valid value
              07    13  0323   786            BEQL    25$                     ; Branch if ok
        20    66    91  0325   787            CMPB    (R6),#DAP$K_IDX         ; Check for valid value
              02    13  0328   788            BEQL    25$                     ; Branch if ok
              52    11  032A   789            BRB     ATT_INVALID             ; Branch on error
                    05  032C   790 25$:       RSB                             ;
                        032D   791 30$:       STORE_FIELD     RFM,1,K_FIX     ; Save record format field
                        0334   792
                        0334   793            ASSUME  DAP$K_UDF EQ 0
                        0334   794            ASSUME  DAP$K_FIX EQ 1
                        0334   795            ASSUME  DAP$K_VAR EQ 2
                        0334   796            ASSUME  DAP$K_VFC EQ 3
                        0334   797            ASSUME  DAP$K_STM EQ 4
                        0334   798            ASSUME  DAP$K_STMLF EQ 5
                        0334   799            ASSUME  DAP$K_STMCR EQ 6
                        0334   800
        06    66    91  0334   801            CMPB    (R6),#DAP$K_STMCR       ; Check for valid value
              45    1A  0337   802            BGTRU   ATT_INVALID             ; Branch if out-of-range
                    05  0339   803            RSB                             ;
                        033A   804 140$:      STORE_FIELD     BSZ,1,K_FIX     ; Save byte size field
                    05  0341   805            RSB                             ;
                        0342   806 150$:      STORE_FIELD     DEV,4,K_EXT     ; Save device characteristics field
                        0349   807            CHECK_MASKS     DEV,4           ; Validate bit options
                    05  0355   808            RSB                             ;
                        0356   809 170$:      STORE_FIELD     LRL,2,K_FIX     ; Save longest record length field
                    05  035D   810            RSB                             ;
                        035E   811 180$:      STORE_FIELD     HBK,4,K_IMG     ; Save highest virtual block number
                    05  0365   812            RSB                             ;  field
                        0366   813 190$:      STORE_FIELD     EBK,4,K_IMG     ; Save end-of-file block number field
                    05  036D   814            RSB                             ;
                        036E   815 200$:      STORE_FIELD     FFB,2,K_FIX     ; Save first free byte in EOF block
                    05  0375   816            RSB                             ;  field
                        0376   817 210$:      STORE_FIELD     SBN,4,K_IMG     ; Save starting logical block number
                    05  037D   818            RSB                             ;  field
                        037E   819
                        037E   820 ;
                        037E   821 ; Branch here on exception condition.
                        037E   822 ;
                        037E   823
                        037E   824 ATT_INVALID:                               ;
            05AB    31  037E   825            BRW     ERROR_INVALID           ; Branch aid
```

```
                         0381          .SBTTL  ACC_MSG - DECODE ACCESS MESSAGE
                         0381
                         0381  ;++
                         0381  ; Decode the operand fields of the Access message.
                         0381  ;--
                         0381
                         0381  ACC_MSG:                                ; Code segment of mainline
                         0381
                         0381  ;
                         0381  ; For optional fields, apply default values as appropriate.
                         0381  ;
                         0381  ; Note: The default value for the DISPLAY field is applied after the ACCFUNC
                         0381  ;       field is processed.
                         0381  ;
                         0381
      42 A9    02    90  0381          MOVB    #DAP$K_FAC_D,DAP$B_FAC(R9)
      43 A9    00    90  0385          MOVB    #DAP$K_SHR_D,DAP$B_SHR(R9)
      48 A9    69    7E  0389          MOVAQ   (R9),DAP$Q_FILESPEC+4(R9) ; Initialize descriptor
      54 A9    69    7E  038D          MOVAQ   (R9),DAP$Q_PASSWORD+4(R9) ; Initialize descriptor
                         0391
                         0391  ;
                         0391  ; Process the access function field (required).
                         0391  ;
                         0391
  58   091D'CF       9E  0391          MOVAB   W^ERROR_FORMAT,R8         ; Specify transfer address on EOM
                         0396          STORE_FIELD     ACCFUNC,1,K_FIX ; Save access function field
                         039D
                         039D          ASSUME  DAP$K_OPEN EQ 1
                         039D          ASSUME  DAP$K_CREATE EQ 2
                         039D          ASSUME  DAP$K_RENAME EQ 3
                         039D          ASSUME  DAP$K_ERASE EQ 4
                         039D          ASSUME  DAP$K_DIR_LIST EQ 6
                         039D          ASSUME  DAP$K_SUBMIT EQ 7
                         039D          ASSUME  DAP$K_EXECUTE EQ 8
                         039D
                         039D          $CASEB  SELECTOR=(R6),-           ; Check for valid value
                         039D                  BASE=#DAP$K_OPEN-         ;
                         039D                  DISPL=<-                  ; Function:
                         039D                          10$-              ;  $OPEN
                         039D                          10$-              ;  $CREATE
                         039D                          20$-              ;  $RENAME
                         039D                          20$-              ;  $ERASE
                         039D                          ERROR_INVALID-    ;  Reserved
                         039D                          20$-              ;  Directory list
                         039D                          10$-              ;  Submit command file
                         039D                          20$-              ;  Execute command file
                         039D                  >                         ;
          0578       31  03B1          BRW     ERROR_INVALID             ; Value out-of-range
            01       B0  03B4  10$:    MOVW    #DAP$M_DSP_ATT,-          ; Apply default DISPLAY value
         4C A9           03B6                  DAP$W_DISPLAY1(R9)        ;  per ACCFUNC value
                         03B8
                         03B8  ;
                         03B8  ; Process the access options and filespec fields (required).
                         03B8  ;
                         03B8
                         03B8  20$:    STORE_FIELD     ACCOPT,1,K_EXT  ; Save access options field
                         03BF          CHECK_MASKS     ACCOPT,1        ; Validate bit options
```

```
                          03C5          STORE_FIELD     FILESPEC,8,K_IMG,<M_DESC>
                          03CC                                                  ; Save descriptor of file
                          03CC                                                  ;  specification string
                          03CC
                          03CC    ;
                          03CC    ; Process the file access and file sharing fields (optional).
                          03CC    ;
                          03CC
      58   0945'CF    9E  03CC          MOVAB   W^EXIT_SUCCESS,R8       ; All done if end-of-message
                          03D1          STORE_FIELD     FAC,1,K_EXT     ; Save file access field
                          03D8          CHECK_MASKS     FAC,1           ; Validate bit options
                          03DE          STORE_FIELD     SHR,1,K_EXT     ; Save file sharing field
                          03E5          CHECK_MASKS     SHR,1           ; Validate bit options
                          03EB
                          03EB    ;
                          03EB    ; Process the display and password fields (optional).
                          03EB    ;
                          03EB
                          03EB          STORE_FIELD     DISPLAY1,2,K_EXT; Save display attributes field
                          03F2          CHECK_MASKS     DISPLAY,2       ; Validate bit options
                          03FA          STORE_FIELD     PASSWORD,8,K_IMG,<M_DESC>
                          0401                                          ; Save descriptor of password string
              68      17  0401          JMP     (R8)                    ; Message syntax is correct
                          0403
                          0403    ;
                          0403    ; Branch here on exception condition.
                          0403    ;
                          0403
                          0403    ACC_INVALID:                          ;
            0526      31  0403          BRW     ERROR_INVALID           ; Branch aid
```

```
                          0406            .SBTTL  CTL_MSG - DECODE CONTROL MESSAGE
                          0406
                          0406    ;++
                          0406    ; Decode the operand fields of the Control message.
                          0406    ;--
                          0406
                          0406    CTL_MSG:                                ; Code segment of mainline
                          0406
                          0406    ;
                          0406    ; For optional fields, apply default values as appropriate.
                          0406    ;
                          0406
     4C A9   69    7E     0406            MOVAQ   (R9),DAP$Q_KEY+4(R9)    ; Initialize descriptor
                          040A
                          040A    ;
                          040A    ; Process the control function field (required).
                          040A    ;
                          040A
58   091D'CF       9E     040A            MOVAB   W^ERROR_FORMAT,R8       ; Specify transfer address on EOM
                          040F            STORE_FIELD     CTLFUNC,1,K_FIX ; Save control function field
                          0416
                          0416            ASSUME  DAP$K_GET_READ EQ 1
                          0416            ASSUME  DAP$K_CONNECT EQ 2
                          0416            ASSUME  DAP$K_UPDATE EQ 3
                          0416            ASSUME  DAP$K_PUT_WRITE EQ 4
                          0416            ASSUME  DAP$K_DELETE EQ 5
                          0416            ASSUME  DAP$K_REWIND EQ 6
                          0416            ASSUME  DAP$K_TRUNCATE EQ 7
                          0416            ASSUME  DAP$K_RELEASE EQ 9
                          0416            ASSUME  DAP$K_FREE EQ 10
                          0416            ASSUME  DAP$K_EXTEND_B EQ 11
                          0416            ASSUME  DAP$K_FLUSH EQ 12
                          0416            ASSUME  DAP$K_FIND EQ 14
                          0416            ASSUME  DAP$K_EXTEND_E EQ 15
                          0416            ASSUME  DAP$K_DISPLAY EQ 16
                          0416            ASSUME  DAP$K_SPACE_FW EQ 17
                          0416            ASSUME  DAP$K_SPACE_BW EQ 18
                          0416
                          0416            $CASEB  SELECTOR=(R6)-          ; Check for valid value
                          0416                    BASE=#DAP$K_GET_READ-   ;
                          0416                    DISPL=<-                ; Function:
                          0416                            10$-            ;  $GET or $READ
                          0416                            10$-            ;  $CONNECT
                          0416                            10$-            ;  $UPDATE
                          0416                            10$-            ;  $PUT or $WRITE
                          0416                            10$-            ;  $DELETE
                          0416                            10$-            ;  $REWIND
                          0416                            10$-            ;  $TRUNCATE
                          0416                            ERROR_INVALID-  ;  Reserved for $MODIFY
                          0416                            10$-            ;  $RELEASE
                          0416                            10$-            ;  $FREE
                          0416                            10$-            ;  $EXTEND (beginning message of seq)
                          0416                            10$-            ;  $FLUSH
                          0416                            ERROR_UNSUPPORT- ; Reserved for $NXTVOL--was defined
                          0416                            10$-            ;  $FIND
                          0416                            10$-            ;  $EXTEND (ending message of seq)
                          0416                            10$-            ;  $DISPLAY
                          0416                     10$-                    ;  $SPACE (forward)
                          0416                     10$-                    ;  $SPACE (backward)
                          0416               >                             ;  Reserved for checkpoint-file function
                          043E                                             ;  Reserved for recovery-get function
                          043E                                             ;  Reserved for recovery-put function
            04EB   31     043E               BRW     ERROR_INVALID         ; Value is out-of-range
                          0441
                          0441      ;
                          0441      ; Process the control menu field (optional).
                          0441      ; Each bit set denotes that its associated field follows in the message.
                          0441      ;
                          0441
                          0441               ASSUME  DAP$V_RAC+1 EQ DAP$V_KEY
                          0441               ASSUME  DAP$V_KEY+1 EQ DAP$V_KRF
                          0441               ASSUME  DAP$V_KRF+1 EQ DAP$V_ROP
                          0441               ASSUME  DAP$V_ROP+2 EQ DAP$V_DISPLAY2
                          0441               ASSUME  DAP$V_DISPLAY2+1 EQ DAP$V_BLKCNT
                          0441
       58  0945'CF   9E   0441      10$:     MOVAB   W^EXIT_SUCCESS,R8     ; All done if end-of-message
                          0446               STORE_FIELD     CTLMENU,2,K_EXT ; Save control menu field
                          044D               CHECK_MASKS     CTLMENU,2     ; Validate bit options
       58  091D'CF   9E   0455               MOVAB   W^ERROR_FORMAT,R8     ; Specify transfer address on EOM
             5C  66  3C   045A               MOVZWL  (R6),AP               ; Copy menu to scratch register
                          045D      CTL_LOOP:
   50  5C  07   00   EA   045D               FFS     #0,#DAP$V_BLKCNT+1,AP,R0; Get position of next bit set
                          0462               $CLRBIT R0,AP                 ; Clear menu bit just found
             F4  AF  9F   0466               PUSHAB  B^CTL_LOOP            ; Push return address on stack
                          0469               $CASEB  SELECTOR=R0-          ; Next field:
                          0469                       DISPL=<-              ;
                          0469                              10$-           ;  RAC
                          0469                              20$-           ;  KEY
                          0469                              30$-           ;  KRF
                          0469                              40$-           ;  ROP
                          0469                              ERROR_FORMAT-  ;  Reserved
                          0469                              60$-           ;  DISPLAY2
                          0469                              70$-           ;  BLKCNT
                          0469                       >                     ;
            04C7   31     047B               BRW     EXIT_SUCCESS          ; Message syntax is correct
                          047E
                          047E      ;
                          047E      ; Process the fields specified in the menu (optional).
                          047E      ;
                          047E
                          047E      10$:     STORE_FIELD     RAC,1,K_FIX   ; Save record access field
                          0485
                          0485               ASSUME  DAP$K_SEQ_ACC EQ 0
                          0485               ASSUME  DAP$K_KEY_ACC EQ 1
                          0485               ASSUME  DAP$K_RFA_ACC EQ 2
                          0485               ASSUME  DAP$K_SEQ_FILE EQ 3
                          0485               ASSUME  DAP$K_BLK_VBN EQ 4
                          0485               ASSUME  DAP$K_BLK_FILE EQ 5
                          0485
          05   66    91   0485               CMPB    (R6),#DAP$K_BLK_FILE  ; Check for value too high
                 3D  1A   0488               BGTRU   CTL_INVALID           ; Branch on error
                     05   048A               RSB                           ;
                          048B      20$:     STORE_FIELD     KEY,8,K_IMG,<M_DESC>
                          0492                                             ; Save descriptor of key string
```

```
           05  0492             RSB                                ;
               0493  30$:       STORE_FIELD     KRF,1,K_FIX        ; Save key of reference field
           05  049A             RSB                                ;
               049B  40$:       STORE_FIELD     ROP,4,K_EXT        ; Save record options field
               04A2             CHECK_MASKS     ROP,4              ; Validate bit options
           05  04AE             RSB                                ;
               04AF  60$:       STORE_FIELD     DISPLAY2,2,K_EXT   ; Save display attributes field
               04B6             CHECK_MASKS     DISPLAY,2          ; Validate bit options
           05  04BE             RSB                                ;
               04BF  70$:       STORE_FIELD     BLKCNT,1,K_FIX     ; Save block count field
           05  04C6             RSB                                ;
               04C7
               04C7  ;
               04C7  ; Branch here on exception condition.
               04C7  ;
               04C7
               04C7  CTL_INVALID:                                  ;
     0462  31  04C7             BRW     ERROR_INVALID              ; Branch aid
```

```
                             04CA              .SBTTL  CON_MSG - DECODE CONTINUE TRANSFER MESSAGE
                             04CA
                             04CA        ;++
                             04CA        ; Decode the operand fields of the Continue Transfer message.
                             04CA        ;--
                             04CA
                             04CA        CON_MSG:                                    ; Code segment of mainline
                             04CA
                             04CA        ;
                             04CA        ; Process the continue transfer function field (required).
                             04CA        ;
                             04CA
        58   091D'CF     9E  04CA                MOVAB   W^ERROR_FORMAT,R8           ; Specify transfer address on EOM
                             04CF                STORE_FIELD     CONFUNC,1,K_FIX     ; Save continue transfer function field
                             04D6
                             04D6                ASSUME  DAP$K_RETRY EQ 1
                             04D6                ASSUME  DAP$K_SKIP_REC EQ 2
                             04D6                ASSUME  DAP$K_ABORT EQ 3
                             04D6                ASSUME  DAP$K_RESUME EQ 4
                             04D6                ASSUME  DAP$K_QUIT EQ 5
                             04D6
                  66     95  04D6                TSTB    (R6)                        ; Branch if value is
                  08     13  04D8                BEQL    CON_INVALID                 ;  too low
            05    66     91  04DA                CMPB    (R6),#DAP$K_QUIT            ;  or
                  03     1A  04DD                BGTRU   CON_INVALID                 ;  too high
                0463     31  04DF                BRW     EXIT_SUCCESS                ; Message syntax is correct
                             04E2
                             04E2        ;
                             04E2        ; Branch here on exception condition.
                             04E2        ;
                             04E2
                             04E2        CON_INVALID:                                ;
                0447     31  04E2                BRW     ERROR_INVALID               ; Branch aid
```

```
                    04E5          .SBTTL  CMP_MSG - DECODE ACCESS COMPLETE MESSAGE
                    04E5
                    04E5  ;++
                    04E5  ; Decode the operand fields of the Access Complete message.
                    04E5  ;--
                    04E5
                    04E5  CMP_MSG:                                ; Code segment of mainline
                    04E5
                    04E5  ;
                    04E5  ; For optional fields, apply default values as appropriate.
                    04E5  ;
                    04E5
                    04E5  ;       <there are no defaults to apply>
                    04E5
                    04E5  ;
                    04E5  ; Process the access complete function field (required).
                    04E5  ;
                    04E5
58   091D'CF   9E   04E5          MOVAB   W^ERROR_FORMAT,R8       ; Specify transfer address on EOM
                    04EA          STORE_FIELD     CMPFUNC,1,K_FIX ; Save access complete function field
                    04F1
                    04F1          ASSUME  DAP$K_CLOSE EQ 1
                    04F1          ASSUME  DAP$K_RESPONSE EQ 2
                    04F1          ASSUME  DAP$K_RESET EQ 3
                    04F1          ASSUME  DAP$K_DISCONN EQ 4
                    04F1          ASSUME  DAP$K_SKIP_FILE EQ 5
                    04F1          ASSUME  DAP$K_CHANGE_B EQ 6
                    04F1          ASSUME  DAP$K_CHANGE_E EQ 7
                    04F1          ASSUME  DAP$K_TERMINATE EQ 8
                    04F1
          66   95   04F1          TSTB    (R6)                    ; Branch if value is
          2B   13   04F3          BEQL    CMP_INVALID             ;  too low
     08   66   91   04F5          CMPB    (R6),#DAP$K_TERMINATE   ;  or
          26   1A   04F8          BGTRU   CMP_INVALID             ;  too high
                    04FA
                    04FA  ;
                    04FA  ; Process the file options field (optional).
                    04FA  ;
                    04FA
58   0945'CF   9E   04FA          MOVAB   W^EXIT_SUCCESS,R8       ; All done if end-of-message
                    04FF          STORE_FIELD     FOP2,4,K_EXT    ; Save file options field
                    0506          CHECK_MASKS     FOP,4           ; Validate bit options
                    0512
                    0512  ;
                    0512  ; Process the CRC checksum field (optional).
                    0512  ;
                    0512
                    0512          STORE_FIELD     CHECK,2,K_FIX   ; Save CRC checksum field
                    0519          $SETBIT #DAP$V_X_CHECK,-        ; Denote field explicitly specified
                    0519                  DAP$B_X_FIELD(R9)       ;  (to distinguish between CRC value
                    051E                                          ;  of zero and none specified)
          68   17   051E          JMP     (R8)                    ; Message syntax is correct
                    0520
                    0520  ;
                    0520  ; Branch here on exception condition.
                    0520  ;
                    0520
```

```
                                   0520  1139 CMP_INVALID:
                       0409   31   0520  1140           BRW     ERROR_INVALID           ; Branch aid
```

```
                          0523                 .SBTTL  DAT_MSG - DECODE DATA MESSAGE
                          0523
                          0523         ;++
                          0523         ; Decode the operand fields of the Data message.
                          0523         ;--
                          0523
                          0523         DAT_MSG:                                        ; Code segment of mainline
                          0523
                          0523         ;
                          0523         ; For optional fields, apply default values as appropriate.
                          0523         ;
                          0523
        48 A9    69    7E 0523                 MOVAQ   (R9),DAP$Q_FILEDATA+4(R9) ; Initialize descriptor
                          0527
                          0527         ;
                          0527         ; Process the record number field (required).
                          0527         ;
                          0527         ; Note: Since there is no menu for the Data message (an unfortunate oversight
                          0527         ;       in the DAP spec), this field must be present. However, it is necessary
                          0527         ;       to distinguish between receiving a null value and a zero value, so that
                          0527         ;       it can be determined whether the RECNUM field overrides the KEY field.
                          0527         ;       To solve this problem, the DAP spec states that a byte count of zero
                          0527         ;       for this image field means that no value has been specified. I.e.,
                          0527         ;       <byte0 = 0> means ignore field, whereas <byte0 = 1 and byte1 = 0> means
                          0527         ;       a value of zero overrides the KEY field value.
                          0527         ;
                          0527
     58    091D'CF     9E 0527                 MOVAB   W^ERROR_FORMAT,R8       ; Specify transfer address on EOM
     5C    5B     01   C1 052C                 ADDL3   #1,R11,AP               ; Mark address of next byte + 1
                          0530                 STORE_FIELD     RECNUM1,4,K_IMG ; Save record number field
           5C     5B   D1 0537                 CMPL    R11,AP                  ; Branch if this image field was
                  05   13 053A                 BEQLU   10$                     ;  exactly one byte long
                          053C                 $SETBIT #DAP$V_X_RECNUM,-       ; Denote field explicitly specified
                          053C                         DAP$B_X_FIELD(R9)       ;  in message
                          0541
                          0541         ;
                          0541         ; Process the file data field (optional for zero length record).
                          0541         ;
                          0541
     58    0945'CF     9E 0541         10$:    MOVAB   W^EXIT_SUCCESS,R8       ; All done if end-of-message
                          0546                 STORE_FIELD     FILEDATA,8,K_ROM,<M_DESC>
                          054D                                                 ; Save descriptor of user data string
                          054D                                                 ;  (the record/block just received)
                  68   17 054D                 JMP     (R8)                    ; Message syntax is correct
```

```
                          054F         .SBTTL  KEY_MSG - DECODE KEY DEFINITION MESSAGE
                          054F
                          054F  ;++
                          054F  ; Decode the operand fields of the Key Definition message.
                          054F  ;--
                          054F
                          054F  KEY_MSG:                                       ; Code segment of mainline
                          054F
                          054F  ;
                          054F  ; For optional fields, apply default values as appropriate.
                          054F  ;
                          054F
      68 A9    69    7E   054F         MOVAQ   (R9),DAP$Q_KNM+4(R9)        ; Initialize descriptor
                          0553
                          0553  ;
                          0553  ; Process the key menu field (optional).
                          0553  ; Each bit set denotes that its associated field follows in the message.
                          0553  ;
                          0553
                          0553         ASSUME  DAP$V_FLG+1 EQ DAP$V_DFL
                          0553         ASSUME  DAP$V_DFL+1 EQ DAP$V_IFL
                          0553         ASSUME  DAP$V_IFL+1 EQ DAP$V_NSG
                          0553         ASSUME  DAP$V_NSG+1 EQ DAP$V_REF
                          0553         ASSUME  DAP$V_REF+1 EQ DAP$V_KNM
                          0553         ASSUME  DAP$V_KNM+1 EQ DAP$V_NUL
                          0553         ASSUME  DAP$V_NUL+1 EQ DAP$V_IAN
                          0553         ASSUME  DAP$V_IAN+1 EQ DAP$V_LAN
                          0553         ASSUME  DAP$V_LAN+1 EQ DAP$V_DAN
                          0553         ASSUME  DAP$V_DAN+1 EQ DAP$V_DTP
                          0553         ASSUME  DAP$V_DTP+1 EQ DAP$V_RVB
                          0553         ASSUME  DAP$V_RVB+2 EQ DAP$V_DVB
                          0553         ASSUME  DAP$V_DVB+1 EQ DAP$V_DBS
                          0553         ASSUME  DAP$V_DBS+1 EQ DAP$V_IBS
                          0553         ASSUME  DAP$V_IBS+1 EQ DAP$V_LVL
                          0553         ASSUME  DAP$V_LVL+1 EQ DAP$V_TKS
                          0553         ASSUME  DAP$V_TKS+1 EQ DAP$V_MRL
                          0553
         58   0945'CF    9E   0553         MOVAB   W^EXIT_SUCCESS,R8           ; All done if end-of-message
                          0558         STORE_FIELD     KEYMENU,4,K_EXT ; Save key definition menu field
                          055F         CHECK_MASKS     KEYMENU,4       ; Validate bit options
         58   091D'CF    9E   056B         MOVAB   W^ERROR_FORMAT,R8           ; Specify transfer address on EOM
              5C    66    D0   0570         MOVL    (R6),AP                     ; Copy menu to scratch register
                          0573  KEY_LOOP:                                      ;
   50    5C    13    00    EA   0573         FFS     #0,#DAP$V_MRL+1,AP,R0       ; Get position of next bit set
                          0578         $CLRBIT R0,AP                       ; Clear menu bit just found
              F4 AF    9F   057C         PUSHAB  B^KEY_LOOP                  ; Push return address on stack
                          057F         $CASEB  SELECTOR=R0-                ; Next field:
                          057F                 DISPL=<-                    ;
                          057F                         10$-                ;  FLG
                          057F                         20$-                ;  DFL
                          057F                         30$-                ;  IFL
                          057F                         40$-                ;  NSG, POS, SIZ
                          057F                         50$-                ;  REF
                          057F                         60$-                ;  KNM
                          057F                         70$-                ;  NUL
                          057F                         80$-                ;  IAN
                          057F                         90$-                ;  LAN
```

```
                        057F  1244                      100$-                 ; DAN
                        057F  1245                      110$-                 ; DTP
                        057F  1246                      120$-                 ; RVB
                        057F  1247                      ERROR_FORMAT-         ; Reserved
                        057F  1248                      140$-                 ; DVB
                        057F  1249                      150$-                 ; DBS
                        057F  1250                      160$-                 ; IBS
                        057F  1251                      170$-                 ; LVL
                        057F  1252                      180$-                 ; TKS
                        057F  1253                      190$-                 ; MRL
                        057F  1254          >                                 ;
         0399     31    05A9  1255          BRW     EXIT_SUCCESS              ; Message syntax is correct
                        05AC  1256
                        05AC  1257  ;
                        05AC  1258  ; Process each field specified in the menu (optional).
                        05AC  1259  ;
                        05AC  1260
                        05AC  1261  10$:    STORE_FIELD     FLG,1,K_EXT       ; Save key options field
                        05B3  1262          CHECK_MASKS     FLG,1             ; Validate bit options
                  05    05B9  1263          RSB                               ;
                        05BA  1264  20$:    STORE_FIELD     DFL,2,K_FIX       ; Save data bucket fill quantity field
                  05    05C1  1265          RSB                               ;
                        05C2  1266  30$:    STORE_FIELD     IFL,2,K_FIX       ; Save index bucket fill quantity field
                  05    05C9  1267          RSB                               ;
                        05CA  1268  40$:    STORE_FIELD     NSG,1,K_FIX       ; Save number of key segments field
      52     66   9A    05D1  1269          MOVZBL  (R6),R2                   ; Use number of segments as loop count
             2C   13    05D4  1270          BEQL    47$                       ; Branch if zero
      08     52   D1    05D6  1271          CMPL    R2,#8                     ; Check for value too high
             28   1A    05D9  1272          BGTRU   49$                       ; Branch on error
 50     4C   A9   3E    05DB  1273          MOVAW   DAP$W_POS(R9),R0          ; Get address of POS array
 51     5C   A9   9E    05DF  1274          MOVAB   DAP$B_SIZ(R9),R1          ; Get address of SIZ array
             07   BB    05E3  1275  45$:    PUSHR   #^M<R0,R1,R2>             ;
                        05E5  1276          STORE_FIELD     POS_TMP,2,K_FIX   ; Find next key segment size field
             01   BA    05EC  1277          POPR    #^M<R0>                   ;
      80     66   B0    05EE  1278          MOVW    (R6),(R0)+                ; Save it in array
             01   BB    05F1  1279          PUSHR   #^M<R0>                   ;
                        05F3  1280          STORE_FIELD     SIZ_TMP,1,K_FIX   ; Find next key segment size field
             07   BA    05FA  1281          POPR    #^M<R0,R1,R2>             ;
      81     66   90    05FC  1282          MOVB    (R6),(R1)+                ; Save it in array
         E1  52   F5    05FF  1283          SOBGTR  R2,45$                    ; Branch if more segments to go
                  05    0602  1284  47$:    RSB                               ;
             7F   11    0603  1285  49$:    BRB     KEY_INVALID               ; Branch aid
                        0605  1286  50$:    STORE_FIELD     REF,1,K_FIX       ; Save key of reference field
                  05    060C  1287          RSB                               ;
                        060D  1288  60$:    STORE_FIELD     KNM,8,K_IMG,<M_DESC>
                        0614  1289                                            ; Save descriptor of key name string
      28     66   91    0614  1290          CMPB    (R6),#40                  ; Check for string too long
             6B   1A    0617  1291          BGTRU   KEY_INVALID               ; Branch on error
      20     66   91    0619  1292          CMPB    (R6),#32                  ; Check for string too long
             66   1A    061C  1293          BGTRU   KEY_INVALID               ; Branch on error
                  05    061E  1294          RSB                               ;
                        061F  1295  70$:    STORE_FIELD     NUL,1,K_FIX       ; Save null key character field
                  05    0626  1296          RSB                               ;
                        0627  1297  80$:    STORE_FIELD     IAN,1,K_FIX       ; Save index area number field
                  05    062E  1298          RSB                               ;
                        062F  1299  90$:    STORE_FIELD     LAN,1,K_FIX       ; Save lowest level index area number
                        0636  1300                                            ;  field
                  05  0636  1301         RSB                                 ;
                      0637  1302 100$:   STORE_FIELD     DAN,1,K_FIX         ; Save data area number field
                  05  063E  1303         RSB                                 ;
                      063F  1304 110$:   STORE_FIELD     DTP,1,K_FIX         ; Save key data type field
                      0646  1305
                      0646  1306         ASSUME  DAP$K_STG EQ 0
                      0646  1307         ASSUME  DAP$K_IN2 EQ 1
                      0646  1308         ASSUME  DAP$K_BN2 EQ 2
                      0646  1309         ASSUME  DAP$K_IN4 EQ 3
                      0646  1310         ASSUME  DAP$K_BN4 EQ 4
                      0646  1311         ASSUME  DAP$K_PAC EQ 5
                      0646  1312         ASSUME  DAP$K_IN8 EQ 6
                      0646  1313         ASSUME  DAP$K_BN8 EQ 7
                      0646  1314
   07      66     91  0646  1315         CMPB    (R6),#DAP$K_BN8             ; Check for value too high
           39     1A  0649  1316         BGTRU   KEY_INVALID                 ; Branch on error
                  05  064B  1317         RSB                                 ;
                      064C  1318 120$:   STORE_FIELD     RVB,4,K_IMG         ; Save root bucket start VBN field
                  05  0653  1319         RSB                                 ;
                      0654  1320 140$:   STORE_FIELD     DVB,4,K_IMG         ; Save first data bucket start VBN field
                  05  065B  1321         RSB                                 ;
                      065C  1322 150$:   STORE_FIELD     DBS,1,K_FIX         ; Save data bucket fill size field
                  05  0663  1323         RSB                                 ;
                      0664  1324 160$:   STORE_FIELD     IBS,1,K_FIX         ; Save index bucket fill size field
                  05  066B  1325         RSB                                 ;
                      066C  1326 170$:   STORE_FIELD     LVL,1,K_FIX         ; Save level of root buckets field
                  05  0673  1327         RSB                                 ;
                      0674  1328 180$:   STORE_FIELD     TKS,1,K_FIX         ; Save total key size field
                  05  067B  1329         RSB                                 ;
                      067C  1330 190$:   STORE_FIELD     MRL,2,K_FIX         ; Save minimum record length to contain
                  05  0683  1331         RSB                                 ;  key field
                      0684  1332
                      0684  1333 ;
                      0684  1334 ; Branch here on exception condition.
                      0684  1335 ;
                      0684  1336
                      0684  1337 KEY_INVALID:                               ;
         02A5     31  0684  1338         BRW     ERROR_INVALID               ; Branch aid
```

```
                      0687          .SBTTL  ALL_MSG - DECODE ALLOCATION MESSAGE
                      0687
                      0687  ;++
                      0687  ; Decode the operand fields of the Allocation message.
                      0687  ;--
                      0687
                      0687  ALL_MSG:                                        ; Code segment of mainline
                      0687
                      0687  ;
                      0687  ; For optional fields, apply default values as appropriate.
                      0687  ;
                      0687
                      0687  ;       <there are no defaults to apply>
                      0687
                      0687  ;
                      0687  ; Process the allocation menu field (optional).
                      0687  ; Each bit set denotes that its associated field follows in the message.
                      0687  ;
                      0687
                      0687          ASSUME  DAP$V_VOL+1 EQ DAP$V_ALN
                      0687          ASSUME  DAP$V_ALN+1 EQ DAP$V_AOP
                      0687          ASSUME  DAP$V_AOP+1 EQ DAP$V_LOC
                      0687          ASSUME  DAP$V_LOC+2 EQ DAP$V_ALQ2
                      0687          ASSUME  DAP$V_ALQ2+1 EQ DAP$V_AID
                      0687          ASSUME  DAP$V_AID+1 EQ DAP$V_BKZ
                      0687          ASSUME  DAP$V_BKZ+1 EQ DAP$V_DEQ2
                      0687
    58  0945'CF   9E  0687          MOVAB   W^EXIT_SUCCESS,R8               ; All done if end-of-message
                      068C          STORE_FIELD     ALLMENU,2,K_EXT         ; Save allocation menu field
                      0693          CHECK_MASKS     ALLMENU,2               ; Validate bit options
    58  091D'CF   9E  069B          MOVAB   W^ERROR_FORMAT,R8               ; Specify transfer address on EOM
        5C    66  3C  06A0          MOVZWL  (R6),AP                         ; Copy menu to scratch register
                      06A3  ALL_LOOP:
50  5C  09    00  EA  06A3          FFS     #0,#DAP$V_DEQ2+1,AP,R0          ; Get position of next bit set
                      06A8          $CLRBIT R0,AP                           ; Clear menu bit just found
        F4 AF     9F  06AC          PUSHAB  B^ALL_LOOP                      ; Push return address on stack
                      06AF          $CASEB  SELECTOR=R0-                    ; Next field:
                      06AF                  DISPL=<-                        ;
                      06AF                          10$-                    ;  VOL
                      06AF                          20$-                    ;  ALN
                      06AF                          30$-                    ;  AOP
                      06AF                          40$-                    ;  LOC
                      06AF                          ERROR_FORMAT-           ;  Reserved
                      06AF                          60$-                    ;  ALQ2
                      06AF                          70$-                    ;  AID
                      06AF                          80$-                    ;  BKZ
                      06AF                          90$-                    ;  DEQ2
                      06AF                  >                               ;
          027D    31  06C5          BRW     EXIT_SUCCESS                    ; Message syntax is correct
                      06C8
                      06C8  ;
                      06C8  ; Process each field specified in the menu (optional).
                      06C8  ;
                      06C8
                      06C8  10$:    STORE_FIELD     VOL,2,K_FIX             ; Save volume number field
                  05  06CF          RSB                                     ;
                      06D0
```

```
                 06D0  1397          ASSUME  DAP$K_ANY EQ 0
                 06D0  1398          ASSUME  DAP$K_CYL EQ 1
                 06D0  1399          ASSUME  DAP$K_LBN EQ 2
                 06D0  1400          ASSUME  DAP$K_VBN EQ 3
                 06D0  1401          ASSUME  DAP$K_RFI EQ 4
                 06D0  1402
                 06D0  1403 20$:     STORE_FIELD     ALN,1,K_FIX      ; Save alignment options field
  03   66    91  06D7  1404          CMPB    (R6),#DAP$K_VBN          ; Check for value too high
       37    1A  06DA  1405          BGTRU   ALL_INVALID              ; Branch on error
             05  06DC  1406          RSB                              ;
                 06DD  1407 30$:     STORE_FIELD     AOP,1,K_EXT      ; Save allocation options field
                 06E4  1408          CHECK_MASKS     AOP,1            ; Validate bit options
             05  06EA  1409          RSB                              ;
                 06EB  1410 40$:     STORE_FIELD     LOC,4,K_IMG      ; Save starting location field
             05  06F2  1411          RSB                              ;
                 06F3  1412 60$:     STORE_FIELD     ALQ2,4,K_IMG     ; Save allocation quantity field
             05  06FA  1413          RSB                              ;
                 06FB  1414 70$:     STORE_FIELD     AID,1,K_FIX      ; Save area identification field
             05  0702  1415          RSB                              ;
                 0703  1416 80$:     STORE_FIELD     BKZ,1,K_FIX      ; Save bucket size field
             05  070A  1417          RSB                              ;
                 070B  1418 90$:     STORE_FIELD     DEQ2,2,K_FIX     ; Save default extension quantity field
             05  0712  1419          RSB                              ;
                 0713  1420
                 0713  1421 ;
                 0713  1422 ; Branch here on exception condition.
                 0713  1423 ;
                 0713  1424
                 0713  1425 ALL_INVALID:                             ;
     0216    31  0713  1426          BRW     ERROR_INVALID            ; Branch aid
```

```
                                0716            .SBTTL  TIM_MSG - DECODE DATE AND TIME MESSAGE
                                0716
                                0716    ;++
                                0716    ; Decode the operand fields of the Date and Time message.
                                0716    ;--
                                0716
                                0716    TIM_MSG:                                        ; Code segment of mainline
                                0716
                                0716    ;
                                0716    ; For optional fields, apply default values as appropriate.
                                0716    ;
                                0716
                                0716    ;       <there are no defaults to apply>
                                0716
                                0716    ;
                                0716    ; Process the date and time menu field (optional).
                                0716    ; Each bit set denotes that its associated field follows in the message.
                                0716    ;
                                0716
                                0716            ASSUME  DAP$V_CDT+1 EQ DAP$V_RDT
                                0716            ASSUME  DAP$V_RDT+1 EQ DAP$V_EDT
                                0716            ASSUME  DAP$V_EDT+1 EQ DAP$V_RVN
                                0716            ASSUME  DAP$V_RVN+1 EQ DAP$V_BDT
                                0716            ASSUME  DAP$V_BDT+1 EQ DAP$V_PDT
                                0716            ASSUME  DAP$V_PDT+1 EQ DAP$V_ADT
                                0716
          58   0945'CF     9E   0716            MOVAB   W^EXIT_SUCCESS,R8               ; All done if end-of-message
                                071B            STORE_FIELD     TIMENU,2,K_EXT          ; Save date and time menu field
                                0722            CHECK_MASKS     TIMENU,2                ; Validate bit options
          58   091D'CF     9E   072A            MOVAB   W^ERROR_FORMAT,R8               ; Specify transfer address on EOM
               5C     66   3C   072F            MOVZWL  (R6),AP                         ; Copy menu to scratch register
                                0732    TIM_LOOP:                                       ;
               53     12   D0   0732            MOVL    #18,R3                          ; Declare size of time (CDT, RDT, etc.)
                                0735                                                    ;  fields containing ASCII strings
     50   5C   07     00   EA   0735            FFS     #0,#DAP$V_ADT+1,AP,R0           ; Get position of next bit set
                                073A            $CLRBIT R0,AP                           ; Clear menu bit just found
                    F1 AF  9F   073E            PUSHAB  B^TIM_LOOP                      ; Push return address on stack
                                0741            $CASEB  SELECTOR=R0-                    ; Next field:
                                0741                    DISPL=<-                        ;
                                0741                            10$-                    ;  CDT
                                0741                            20$-                    ;  RDT
                                0741                            30$-                    ;  EDT
                                0741                            40$-                    ;  RVN
                                0741                            50$-                    ;  BDT
                                0741                            60$-                    ;  PDT
                                0741                            70$-                    ;  ADT
                                0741                    >
                  01EF     31   0753            BRW     EXIT_SUCCESS                    ; Message syntax is correct
                                0756
                                0756    ;
                                0756    ; Process each field specified in the menu (optional).
                                0756    ;
                                0756
                                0756    10$:    STORE_FIELD     CDT,8,K_FIX,<M_SRCR3!M_DESC>
                                075D                                                    ; Save descriptor of creation
                                075D                                                    ;  date and time string
                    35     11   075D            BRB     100$                            ;
```

```
                                075F   1485 20$:     STORE_FIELD     RDT,8,K_FIX,<M_SRCR3!M_DESC>
                                0766   1486                                            ; Save descriptor of revision
                                0766   1487                                            ;  date and time string
                     2C    11   0766   1488          BRB     100$                      ;
                                0768   1489 30$:     STORE_FIELD     EDT,8,K_FIX,<M_SRCR3!M_DESC>
                                076F   1490                                            ; Save descriptor of expiration
                                076F   1491                                            ;  date and time string
                     23    11   076F   1492          BRB     100$                      ;
                                0771   1493 40$:     STORE_FIELD     RVN,2,K_FIX       ; Save revision number field
                           05   0778   1494          RSB
                                0779   1495 50$:     STORE_FIELD     BDT,8,K_FIX,<M_SRCR3!M_DESC>
                                0780   1496                                            ; Save descriptor of backup
                                0780   1497                                            ;  date and time string
                     12    11   0780   1498          BRB     100$                      ;
                                0782   1499 60$:     STORE_FIELD     PDT,8,K_FIX,<M_SRCR3!M_DESC>
                                0789   1500                                            ; Save descriptor of physical creation
                                0789   1501                                            ;  date and time string
                     09    11   0789   1502          BRB     100$                      ;
                                078B   1503 70$:     STORE_FIELD     ADT,8,K_FIX,<M_SRCR3!M_DESC>
                                0792   1504                                            ; Save descriptor of accessed
                                0792   1505                                            ;  date and time string
                     00    11   0792   1506          BRB     100$                      ;
                                0794   1507
                                0794   1508 ;+
                                0794   1509 ; This sequence converts an ASCII time string into a 64-bit VMS binary time
                                0794   1510 ; value. Note that the 64-bit result is stored in the quadword descriptor
                                0794   1511 ; pointing to the time string on input.
                                0794   1512 ;
                                0794   1513 ; DAP defines network standard time as an 18 byte counted ASCII string in the
                                0794   1514 ; format "dd-mmm-yybhh:mm:ss", whereas VMS uses a 23-byte ASCII string in the
                                0794   1515 ; format "dd-mmm-yyyybhh:mm:ss.cc".
                                0794   1516 ;-
                                0794   1517
            5E       20    C2   0794   1518 100$:    SUBL2   #<24+8>,SP                ; Allocate space from stack
       18 AE         17    D0   0797   1519          MOVL    #<18+2+3>,24(SP)          ; Form descriptor of buffer to receive
       1C AE         5E    D0   079B   1520          MOVL    SP,28(SP)                 ;  altered ASCII string
  6E   04 B6         07    28   079F   1521          MOVC3   #7,@4(R6),(SP)            ; Copy bytes 1-7 of input string
            36       61    91   07A4   1522          CMPB    (R1),#^A\6\               ; Compare decade against base decade
                     07    1A   07A7   1523          BGTRU   110$                      ; Branch if '70 - '99
                                07A9   1524                                            ; Else it's '00 - '69
       83      3032  8F    B0   07A9   1525          MOVW    #^A\20\,(R3)+             ; Insert missing century digits
                     05    11   07AE   1526          BRB     120$                      ; Continue
       83      3931  8F    B0   07B0   1527 110$:    MOVW    #^A\19\,(R3)+             ; Insert missing century digits
       63      61    0B    28   07B5   1528 120$:    MOVC3   #11,(R1),(R3)             ; Copy bytes 8-18 of input string
  83       2030302E  8F    D0   07B9   1529          MOVL    #^A\.00 \,(R3)+           ; Add hundredths of second digits
                                07C0   1530                                            ; Note R3 now points to time descriptor
       03 AE         20    8A   07C0   1531          BICB2   #^X20,3(SP)               ; Upcase 3-digit month string
       04 AE         20    8A   07C4   1532          BICB2   #^X20,4(SP)               ;  because $BINTIM objects to
       05 AE         20    8A   07C8   1533          BICB2   #^X20,5(SP)               ;  lowercase month
                                07CC   1534          $BINTIM_S-                        ; Convert ASCII time to binary time
                                07CC   1535                  TIMBUF=(R3)-              ;  Address of descriptor of ASCII string
                                07CC   1536                  TIMADR=(R6)               ;  Address of 64-bit result value
            5E       20    C0   07D7   1537          ADDL2   #<24+8>,SP                ; Deallocate space from the stack
               01    50    E9   07DA   1538          BLBC    R0,TIM_INVALID            ; Branch on conversion error
                           05   07DD   1539          RSB                               ; Exit
                                07DE   1540
                                07DE   1541 ;
```

```
                   07DE  ; Branch here on exception condition.
                   07DE  ;
                   07DE
                   07DE  TIM_INVALID:
014B      31       07DE          BRW     ERROR_INVALID           ; Branch aid
```

```
                           07E1          .SBTTL  PRO_MSG - DECODE PROTECTION MESSAGE
                           07E1
                           07E1  ;++
                           07E1  ; Decode the operand fields of the Protection message.
                           07E1  ;--
                           07E1
                           07E1  PRO_MSG:                                        ; Code segment of mainline
                           07E1
                           07E1  ;
                           07E1  ; For optional fields, apply default values as appropriate.
                           07E1  ;
                           07E1
        4C A9   69     7E  07E1          MOVAQ   (R9),DAP$Q_OWNER+4(R9)  ; Initialize descriptor
                           07E5
                           07E5  ;
                           07E5  ; Process the protection menu field (optional).
                           07E5  ; Each bit set denotes that its associated field follows in the message.
                           07E5  ;
                           07E5
                           07E5          ASSUME  DAP$V_OWNER+1 EQ DAP$V_PROSYS
                           07E5          ASSUME  DAP$V_PROSYS+1 EQ DAP$V_PROOWN
                           07E5          ASSUME  DAP$V_PROOWN+1 EQ DAP$V_PROGRP
                           07E5          ASSUME  DAP$V_PROGRP+1 EQ DAP$V_PROWLD
                           07E5
     58   0945'CF      9E  07E5          MOVAB   W^EXIT_SUCCESS,R8               ; All done if end-of-message
                           07EA          STORE_FIELD     PROMENU,2,K_EXT ; Save proection menu field
                           07F1          CHECK_MASKS     PROMENU,2       ; Validate bit options
     58   091D'CF      9E  07F9          MOVAB   W^ERROR_FORMAT,R8               ; Specify transfer address on EOM
          5C    66     3C  07FE          MOVZWL  (R6),AP                         ; Copy menu to scratch register
                           0801  PRO_LOOP:
 50  5C   05    00     EA  0801          FFS     #0,#DAP$V_PROWLD+1,AP,R0; Get position of next bit set
                           0806          $CLRBIT R0,AP                           ; Clear menu bit just found
             F4 AF     9F  080A          PUSHAB  B^PRO_LOOP                      ; Push return address on stack
                           080D          $CASEB  SELECTOR=R0-                    ; Next field:
                           080D                  DISPL=<-                        ;
                           080D                          10$-                    ;  OWNER
                           080D                          20$-                    ;  PROSYS
                           080D                          30$-                    ;  PROOWN
                           080D                          40$-                    ;  PROGRP
                           080D                          50$-                    ;  PROWLD
                           080D                  >                               ;
             0127      31  081B          BRW     EXIT_SUCCESS                    ; Message syntax is correct
                           081E
                           081E  ;
                           081E  ; Process each field specified in the menu (optional).
                           081E  ;
                           081E
                           081E  10$:    STORE_FIELD     OWNER,8,K_IMG,<M_DESC>
                           0825                                                  ; Save descriptor of file owner string
        28      66     91  0825          CMPB    (R6),#40                        ; Declare an error if owner string
                2C     1A  0828          BGTRU   PRO_INVALID                     ;  is too long
                       05  082A          RSB
                           082B  20$:    STORE_FIELD     PROSYS,2,K_EXT  ; Save system protection field
                19     11  0832          BRB     100$                            ;
                           0834  30$:    STORE_FIELD     PROOWN,2,K_EXT  ; Save owner protection field
                10     11  083B          BRB     100$                            ;
                           083D  40$:    STORE_FIELD     PROGRP,2,K_EXT  ; Save group protection field
```

```
         07   11  0844          BRB     100$                    ;
                  0846  50$:    STORE_FIELD     PROWLD,2,K_EXT  ; Save world protection field
                  084D
                  084D  ;
                  084D  ; Perform common validity checks on data in the protection field being
                  084D  ; processed.
                  084D  ;
                  084D
                  084D  100$:   CHECK_MASKS     PROTECT,2       ; Validate bit options
              05  0855          RSB                             ;
                  0856
                  0856  ;
                  0856  ; Branch here on exception condition.
                  0856  ;
                  0856
                  0856  PRO_INVALID:                            ;
       00D3   31  0856          BRW     ERROR_INVALID           ; Branch aid
```

```
                          0859          .SBTTL  NAM_MSG - DECODE NAME MESSAGE
                          0859
                          0859  ;++
                          0859  ; Decode the operand fields of the Name message.
                          0859  ;--
                          0859
                          0859  NAM_MSG:                                  ; Code segment of mainline
                          0859
                          0859  ;
                          0859  ; For optional fields, apply default values as appropriate.
                          0859  ;
                          0859
     48 A9    69     7E   0859          MOVAQ   (R9),DAP$Q_NAMESPEC+4(R9) ; Initialize descriptor
                          085D
                          085D  ;
                          085D  ; Process the name type field (required).
                          085D  ;
                          085D
  58    091D'CF      9E   085D          MOVAB   W^ERROR_FORMAT,R8         ; Specify transfer address on EOM
                          0862          STORE_FIELD     NAMETYPE,1,K_FIX; Save the name type field
                          0869
                          0869  ;
                          0869  ; Process the name field (optional).
                          0869  ;
                          0869
  58    0945'CF      9E   0869          MOVAB   W^EXIT_SUCCESS,R8         ; All done if end-of-message
                          086E          STORE_FIELD     NAMESPEC,8,K_IMG,<M_DESC>
                          0875                                            ; Save descriptor of name
                          0875                                            ;  specification string
     80 8F    66     91   0875          CMPB    (R6),#128                 ; Check for string too long
              02     1A   0879          BGTRU   NAM_INVALID               ; Branch on error
              68     17   087B          JMP     (R8)                      ; Message syntax is correct
                          087D
                          087D  ;
                          087D  ; Branch here on exception condition.
                          087D  ;
                          087D
                          087D  NAM_INVALID:                              ;
            00AC     31   087D          BRW     ERROR_INVALID             ; Branch aid
                          0880
```

```
0880          .SBTTL  STORE_FIELD - STORE NEXT FIELD ROUTINES
0880
0880    ;++
0880    ; Functional Description:
0880    ;
0880    ;       STORE_FIELD invoked from the STORE_FIELD macro results in the execution
0880    ;       of one of the following routines:
0880    ;
0880    ;       STORE_EXT interprets the next field of the DAP message as an
0880    ;       extensible field of 1 to 16 bytes and stores the data portion of
0880    ;       the field in the designated field of the DAP control block.
0880    ;
0880    ;       STORE_FIX interprets the next field of the DAP message as a
0880    ;       fixed length field of 1 to 255 bytes and stores the string in the
0880    ;       designated field of the DAP control block.
0880    ;
0880    ;       STORE_IMG interprets the next field of the DAP message as an
0880    ;       image field of 1 to 256 bytes and stores the data portion of the
0880    ;       field in the designated field of the DAP control block.
0880    ;
0880    ;       STORE_ROM interprets the next field of the DAP message as a
0880    ;       binary field of 1 to 65535 bytes consisting of the rest of the
0880    ;       message and stores the string in the designated field of the DAP
0880    ;       control block.
0880    ;
0880    ; Calling Sequence:
0880    ;
0880    ;       BSBW    STORE_FIELD
0880    ;
0880    ; Input Parameters:
0880    ;
0880    ;       R3      Size in bytes of source field iff V_SRCR3 set and field is in
0880    ;               fixed length format
0880    ;       R8      Address of routine to execute if end-of-message encountered
0880    ;       R9      Address of DAP control block
0880    ;       R10     Address of last byte + 1 of DAP message being parsed
0880    ;       R11     Address of next byte of DAP message being parsed
0880    ;
0880    ;       In-line coded arguments:
0880    ;
0880    ;       Byte0   Size in bytes of the destination field in DAP control block
0880    ;       Byte1   Offset of destination field in DAP control block
0880    ;       Byte2   DAP field identifier (used to build DAP status code on error)
0880    ;       Byte3   Control byte to direct processing of DAP field:
0880    ;               Bits 0-3:               ; Format of source field:
0880    ;                       K_EXT=  ^X00    ;  Extensible field format
0880    ;                       K_FIX=  ^X01    ;  Fixed length field format
0880    ;                       K_IMG=  ^X02    ;  Image field format
0880    ;                       K_ROM=  ^X03    ;  Rest-of-message field format
0880    ;               Bits 4-7:               ; Field processing flags:
0880    ;                       M_DESC= ^X10    ;  Store only descriptor of SRC field
0880    ;                       M_TRUNC=^X20    ;  Truncate extra bytes if SRC field
0880    ;                                       ;  size is larger than DST field size
0880    ;                       M_SRCR3=^X40    ;  Size of SRC field is in R3
0880    ;                                       ;  (applicable only if K_FIX specified)
0880    ;
0880    ; Implicit Inputs:
```

```
                        0880  ;
                        0880  ;       None
                        0880  ;
                        0880  ; Output Parameters:
                        0880  ;
                        0880  ;       R0-R5   Destroyed
                        0880  ;       R6      Address of destination field in DAP control block
                        0880  ;       R7      Field ID value
                        0880  ;       R8-R10  Unchanged
                        0880  ;       R11     Updated next byte pointer
                        0880  ;
                        0880  ; Implicit Outputs:
                        0880  ;
                        0880  ;       The specified field of the DAP control block is updated.
                        0880  ;
                        0880  ; Completion Codes:
                        0880  ;
                        0880  ;       None
                        0880  ;
                        0880  ; Side Effects:
                        0880  ;
                        0880  ;       If end-of-message is encountered, control is given to the specified
                        0880  ;       action routine.
                        0880  ;
                        0880  ;       If a parse error is detected, control is given to an appropriate
                        0880  ;       error routine.
                        0880  ;
                        0880  ;       An exception exit described above, leaves the return address on the
                        0880  ;       stack.
                        0880  ;
                        0880  ;--
                        0880
                        0880 STORE_FIELD:                                   ; Entry point
                        0880
                        0880  ;
                        0880  ; Obtain the in-line coded arguments, check for end-of-message, and transfer
                        0880  ; control to the appropriate routine.
                        0880  ;
                        0880
             50   6E  D0 0880          MOVL    (SP),R0                  ; Get address of in-line arguments
             55   80  9A 0883          MOVZBL  (R0)+,R5                 ; Get DST field size
             56   80  9A 0886          MOVZBL  (R0)+,R6                 ; Get DST field offset
             56   59  C0 0889          ADDL2   R9,R6                    ; Compute DST field address
             57   80  9A 088C          MOVZBL  (R0)+,R7                 ; Get DAP field ID value
             52   80  9A 088F          MOVZBL  (R0)+,R2                 ; Get control byte value
             6E   50  D0 0892          MOVL    R0,(SP)                  ; Bump return address past argument list
             5A   5B  D1 0895          CMPL    R11,R10                  ; Is there at least one byte left?
                  11  18 0898          BGEQ    10$                      ; Branch if end-of-message
51   52   04   00  EF 089A          EXTZV   #0,#4,R2,R1              ; Get index of routine
                         089F          $CASEB  SELECTOR=R1-             ; Dispatch on field format:
                         089F                  DISPL=<-                 ;
                         089F                          STORE_EXT-       ;  Extensible
                         089F                          STORE_FIX-       ;  Fixed length
                         089F                          STORE_IMG-       ;  Image
                         089F                          STORE_ROM-       ;  Rest-of-message
                         089F                  >                        ;
             68   17 08AB 10$:    JMP     (R8)                     ; Jump to designated EOM routine
```

```
                          08AD  1779          .SBTTL  STORE_EXT - STORE EXTENSIBLE FIELD
                          08AD  1780
                          08AD  1781  ;++
                          08AD  1782  ; This routine interprets the next field of the DAP message as an extensible
                          08AD  1783  ; field of 1 to 16 bytes where bit7 of each byte determines whether to
                          08AD  1784  ; continue (1) the field to the next byte or to terminate (0) the field.
                          08AD  1785  ; First, the source field is compressed in a work area (i.e., bit7 of each byte
                          08AD  1786  ; is discarded and the remaining bits are squeezed together). Then, the
                          08AD  1787  ; compressed string is copied to the specified destination field in the DAP
                          08AD  1788  ; control block.
                          08AD  1789  ;--
                          08AD  1790
                          08AD  1791          ASSUME  DAP$K_TEMP GE 16
                          08AD  1792
                          08AD  1793  STORE_EXT:                                ; Code segment of STORE_FIELD
       54    0090 C9   9E 08AD  1794          MOVAB   DAP$L_TEMP(R9),R4         ; Get address of scratch work area
                  50   D4 08B2  1795          CLRL    R0                        ; Initialize bit position index
                  53   D4 08B4  1796          CLRL    R3                        ; Initialize byte count
            5A    5B   D1 08B6  1797  10$:    CMPL    R11,R10                   ; Error if end-of-message is reached
                  62   18 08B9  1798          BGEQ    ERROR_FORMAT              ;  before end-of-field is reached
                  53   D6 08BB  1799          INCL    R3                        ; Increment byte count
            10    53   D1 08BD  1800          CMPL    R3,#16                    ; Branch if SRC field is longer than
                  5B   1A 08C0  1801          BGTRU   ERROR_FORMAT              ;  scratch work area
  64   07   50    6B   F0 08C2  1802          INSV    (R11),R0,#7,(R4)          ; Copy lower 7 bits of next byte
            50    07   C0 08C7  1803          ADDL2   #7,R0                     ; Update bit position index
         E8 8B    07   E0 08CA  1804          BBS     #7,(R11)+,10$             ; Loop if field extends to next byte
  64   53   50    00   F0 08CE  1805          INSV    #0,R0,R3,(R4)             ; Zero fill rest of SRC field
                          08D3  1806                                            ;  (1 bit for each byte compressed)
                  28   11 08D3  1807          BRB     MOVE_FIELD                ; Copy string to DST field
```

```
                              08D5               .SBTTL  STORE_FIX - STORE FIXED LENGTH FIELD
                              08D5
                              08D5      ;++
                              08D5      ; This routine interprets the next field of the DAP message as a fixed length
                              08D5      ; field of 1 to 255 bytes and copies the string to the specified field in the
                              08D5      ; DAP control block.
                              08D5      ;--
                              08D5
                              08D5      STORE_FIX:                                     ; Code segment of STORE_FIELD
     03 52      06      E0    08D5              BBS     #V_SRCR3,R2,10$                ; Branch if SRC field size is in r3
        53      55      D0    08D9              MOVL    R5,R3                          ; DST field size = SRC field size
        01      53      D1    08DC      10$:    CMPL    R3,#1                          ; Branch if field is longer than
                07      1A    08DF              BGTRU   STORE_IMG1                     ;  one byte
        66      8B      90    08E1              MOVB    (R11)+,(R6)                    ; Store field in DAP control block
                        05    08E4              RSB                                    ; Exit
```

```
                               08E5          .SBTTL  STORE_IMG - STORE IMAGE FIELD
                               08E5
                               08E5    ;++
                               08E5    ; This routine interprets the next field of the DAP message as an image field
                               08E5    ; of 1 to 256 bytes where the first byte contains a count of the number of
                               08E5    ; data bytes to follow. The data portion of the field is copied to the
                               08E5    ; specified field of the DAP control block.
                               08E5    ;--
                               08E5
                               08E5    STORE_IMG:                                       ; Code segment of STORE_FIELD
                 53     8B  9A 08E5            MOVZBL  (R11)+,R3                        ; Get byte count of SRC field
                               08E8    STORE_IMG1:                                      ;
                 54     5B  D0 08E8            MOVL    R11,R4                           ; Copy address of data string
        000C 5B  53     5A  F1 08EB            ACBL    R10,R3,R11,MOVE_FIELD            ; Ok if <R3+R11> LEQ <R10>
                        2A  11 08F1            BRB     ERROR_FORMAT                     ; Error if not enough bytes in
                               08F3                                                     ;  message to contain field
```

```
                                    08F3  1842                  .SBTTL  STORE_ROM - STORE REST OF MESSAGE
                                    08F3  1843
                                    08F3  1844  ;++
                                    08F3  1845  ; This routine interprets the next field of the DAP message as a binary field
                                    08F3  1846  ; of 1 to 65535 bytes consisting of the rest of the message. The string is
                                    08F3  1847  ; copied to the specified field of the DAP control block.
                                    08F3  1848  ;--
                                    08F3  1849
                                    08F3  1850  STORE_ROM:                                      ; Code segment of STORE_FIELD
                 53    5A    5B    C3  08F3  1851          SUBL3   R11,R10,R3              ; Compute SRC field size
                       54    5B    D0  08F7  1852          MOVL    R11,R4                  ; Copy SRC field address
                       5B    5A    D0  08FA  1853          MOVL    R10,R11                 ; Advance next byte pointer to EOM
                                    08FD  1854
                                    08FD  1855  ;
                                    08FD  1856  ; <R3,R4> contains descriptor of SRC field, and
                                    08FD  1857  ; <R5,R6> contains descriptor of DST field.
                                    08FD  1858  ;
                                    08FD  1859
                                    08FD  1860  MOVE_FIELD:                                     ; Copy SRC field to DST field with
                                    08FD  1861                                                  ;  zero fill
                    18 52    04    E0  08FD  1862          BBS     #V_DESC,R2,DESCRIPTOR   ; Branch if only descriptor desired
       66    55    00 64    53    2C  0901  1863          MOVC5   R3,(R4),#0,R5,(R6)      ; Move field to DAP control block
                             0F    1B  0907  1864          BLEQU   20$                     ; Done if all SRC bytes are copied
                                    0909  1865                                          ;  (i.e., SRC size LEQU DST size)
                       52    6E    D0  0909  1866          MOVL    (SP),R2                 ; Get address of control flag
                                    090C  1867                                          ;  parameter + 1 (i.e., return address)
              07 FF A2    05    E0  090C  1868          BBS     #V_TRUNC,-1(R2),20$     ; Done if extra bytes are to be
                                    0911  1869                                          ;  truncated; note:
                                    0911  1870                                          ;  R0 = # unmoved bytes
                                    0911  1871                                          ;  R1 = address of unmoved string
                             81    95  0911  1872  10$:    TSTB    (R1)+                   ; Error if any unmoved bytes are
                             23    12  0913  1873          BNEQ    ERROR_UNSUPPORT         ;  non-zero
                       F9 50    F5  0915  1874          SOBGTR  R0,10$                  ; Continue until all extra bytes
                                    0918  1875                                          ;  are checked
                                   05  0918  1876  20$:    RSB                             ; Exit
                                    0919  1877  DESCRIPTOR:                                     ; DST field is a descriptor
                       66    53    7D  0919  1878          MOVQ    R3,(R6)                 ; Store only quadword descriptor
                                   05  091C  1879          RSB                             ;  of SRC field and exit
```

```
                  091D          .SBTTL  ERROR AND SUCCESS EXIT ROUTINES
                  091D
                  091D  ;++
                  091D  ; Message parse has failed.
                  091D  ; Build DAP Status message and exit to caller.
                  091D  ;--
                  091D
                  091D  ERROR_FORMAT:                                   ; Format of message in incorrect
         08   90  091D          MOVB    #DAP$_FORMAT,-                  ; Return MACCODE value
      1B A9       091F                  DAP$B_DCODE_MAC(R9)             ;
01    10 A9   D1  0921          CMPL    DAP$Q_MSG_BUF2(R9),#1           ; Check for one-byte message
         15   12  0925          BNEQ    ERROR_COMMON                    ; Take common path if not
   57    08   9A  0927          MOVZBL  #DAP$_FLAGS,R7                  ; Change to flags field ID code
                  092A                                                  ;  because format error was caused
                  092A                                                  ;  by no flags field in message
         10   11  092A          BRB     ERROR_COMMON                    ; Take common path
                  092C  ERROR_INVALID:                                  ; Field of message has invalid value
         09   90  092C          MOVB    #DAP$_INVALID,-                 ; Return MACCODE value
      1B A9       092E                  DAP$B_DCODE_MAC(R9)             ;
         0A   11  0930          BRB     ERROR_COMMON                    ; Take common path
                  0932  ERROR_SYNC:                                     ; Message received is out-of-sequence
         0A   90  0932          MOVB    #DAP$_MSG_SYNC,-                ; Return MACCODE value
      1B A9       0934                  DAP$B_DCODE_MAC(R9)             ;
         04   11  0936          BRB     ERROR_COMMON                    ; Take common path
                  0938  ERROR_UNSUPPORT:                                ; Field of message has unsupported value
         02   90  0938          MOVB    #DAP$_UNSUPPORT,-               ; Return MACCODE value
      1B A9       093A                  DAP$B_DCODE_MAC(R9)             ;
                  093C  ERROR_COMMON:                                   ; Common error exit sequence
19 A9    57   90  093C          MOVB    R7,DAP$B_DCODE_FID(R9)          ; Return ID of field in error
      18 A9   94  0940          CLRB    DAP$L_DCODE_STS(R9)             ; Indicate failure
         16   11  0943          BRB     EXIT_COMMON                     ; Join common exit code
                  0945
                  0945  ;++
                  0945  ; Message parse has been successful so far, ...
                  0945  ; Make additional validity checks.
                  0945  ;--
                  0945
                  0945  EXIT_SUCCESS:                                   ; Enter here on successful parse
   57    00   9A  0945          MOVZBL  #DAP$_UNKNOWN,R7                ; Set field ID to 'unknown'
   5A    5B   D1  0948          CMPL    R11,R10                         ; Branch if there are any unparsed
         D0   12  094B          BNEQ    ERROR_FORMAT                    ;  bytes left in DAP message
50    1A A9   9A  094D          MOVZBL  DAP$B_DCODE_MSG(R9),R0          ; Get DAP message type
1C A9    50   E1  0951          BBC     R0,DAP$L_MSG_MASK(R9),-         ; Branch if this is not a valid
         DC       0955                  ERROR_SYNC                      ;  message to receive
                  0956
                  0956  ;
                  0956  ; Check for system specific fields in message header.
                  0956  ;
                  0956
      38 A9   D5  0956          TSTL    DAP$Q_SYSPEC(R9)                ; Any system specific fields?
         4A   12  0959          BNEQ    SSP_MINI_MSG                    ; If yes, process them
                  095B
                  095B  ;
                  095B  ; Update message descriptors in DAP control block.
                  095B  ;
                  095B
                  095B  EXIT_COMMON:                                    ; Common exit sequence
```

```
              14 A9   C3  095B       SUBL3   DAP$Q_MSG_BUF2+4(R9),-  ; Compute size of message just parsed
        10 A9    5A       095E               R10,DAP$Q_MSG_BUF2(R9)  ;  and store it in descriptor
        0C A9    5A   D0  0961       MOVL    R10,DAP$Q_MSG_BUF1+4(R9); Store address of next (blocked)
                          0965                                       ;  message in buffer to parse
              10 A9   C2  0965       SUBL2   DAP$Q_MSG_BUF2(R9),-    ; Store size of next (blocked)
              08 A9       0968               DAP$Q_MSG_BUF1(R9)      ;  message in buffer to parse
        50    18 A9   D0  096A       MOVL    DAP$L_DCODE_STS(R9),R0  ; Get return status code
                      04  096E       RET                             ; Return to caller
```

```
                  096F          .SBTTL  CHECK_MASKS - VALIDATE FIELD BIT OPTIONS
                  096F
                  096F  ;++
                  096F  ; Functional Description:
                  096F  ;
                  096F  ;       CHECK_MASKS invoked from the CHECK_MASKS macro examines the designated
                  096F  ;       field for invalid and unsupported bits set.
                  096F  ;
                  096F  ; Calling Sequence:
                  096F  ;
                  096F  ;       BSBW    CHECK_MASKS
                  096F  ;
                  096F  ; Input Parameters:
                  096F  ;
                  096F  ;       R6      Address of designated field in DAP control block
                  096F  ;       R7      Field ID value
                  096F  ;
                  096F  ;       In-line coded arguments:
                  096F  ;
                  096F  ;       Byte0   Size in bytes of the designated field in DAP control block
                  096F  ;       Byten   Mask of invalid bits (1-4 bytes; size specified in byte0)
                  096F  ;       Bytem   Mask of unsupported bits (1-4 bytes; size specified in byte0)
                  096F  ;
                  096F  ; Implicit Inputs:
                  096F  ;
                  096F  ;       None
                  096F  ;
                  096F  ; Output Parameters:
                  096F  ;
                  096F  ;       R0-R1   Destroyed
                  096F  ;       R6-R7   Unchanged
                  096F  ;
                  096F  ; Implicit Outputs:
                  096F  ;
                  096F  ;       The specified field of the DAP control block is validated.
                  096F  ;
                  096F  ; Completion Codes:
                  096F  ;
                  096F  ;       None
                  096F  ;
                  096F  ; Side Effects:
                  096F  ;
                  096F  ;       If any invalid or unsupported bits are set, control is given to an
                  096F  ;       appropriate error routine.
                  096F  ;
                  096F  ;       An exception exit described above, leaves the return address on the
                  096F  ;       stack.
                  096F  ;
                  096F  ;--
                  096F
                  096F  CHECK_MASKS:                                    ; Entry point
   50    6E   D0  096F          MOVL    (SP),R0                         ; Get address of in-line arguments
   51    80   9A  0972          MOVZBL  (R0)+,R1                        ; Get DST field size
                  0975          $CASEB  SELECTOR=R1-                    ; Dispatch on field size:
                  0975                  BASE=#1-                        ;
                  0975                  DISPL=<-                        ;
                  0975                          10$-                    ;  1-byte
                  0975  2004                        20$-            ;  2-bytes
                  0975  2005                        30$-            ;  Error
                  0975  2006                        40$-            ;  4-bytes
                  0975  2007                      >                 ;
          9A  11  0981  2008 30$:     BRB     ERROR_FORMAT          ; Value is out-of-range
    80    66  93  0983  2009 10$:     BITB    (R6),(R0)+            ; Check for invalid bits
          A4  12  0986  2010          BNEQ    ERROR_INVALID         ; Branch on error
    80    66  93  0988  2011          BITB    (R6),(R0)+            ; Check for unsupported bits
          12  11  098B  2012          BRB     50$                   ; Join common code
    80    66  B3  098D  2013 20$:     BITW    (R6),(R0)+            ; Check for invalid bits
          9A  12  0990  2014          BNEQ    ERROR_INVALID         ; Branch on error
    80    66  B3  0992  2015          BITW    (R6),(R0)+            ; Check for unsupported bits
          08  11  0995  2016          BRB     50$                   ; Join common code
    80    66  D3  0997  2017 40$:     BITL    (R6),(R0)+            ; Check for invalid bits
          90  12  099A  2018          BNEQ    ERROR_INVALID         ; Branch on error
    80    66  D3  099C  2019          BITL    (R6),(R0)+            ; Check for unsupported bits
          97  12  099F  2020 50$:     BNEQ    ERROR_UNSUPPORT       ; Branch on error
    6E    50  D0  09A1  2021          MOVL    R0,(SP)               ; Bump return address past argument list
              05  09A4  2022          RSB                           ; Exit
```

```
                        09A5          .SBTTL  SSP_MINI_MSG - DECODE SYSTEM SPECIFIC FIELD
                        09A5
                        09A5  ;++
                        09A5  ; Decode the system specific field found in the message header.
                        09A5  ; Treat it as the operand portion of a mini-message that has a menu field
                        09A5  ; and related fields.
                        09A5  ;--
                        09A5
                        09A5  SSP_MINI_MSG:                           ; Code segment of mainline
                 5A  DD 09A5          PUSHL   R10                     ; Save end-of-message + 1 address
         5A   38 A9  7D 09A7          MOVQ    DAP$Q_SYSPEC(R9),R10    ; R10 = size of syspec field
                        09AB                                          ; R11 = address of start-of-field
            5A   5B  C0 09AB          ADDL2   R11,R10                 ; R10 = address of end-of-field + 1
                        09AE          $ZERO_FILL-                     ; Zero system specific work area
                        09AE                  DST=DAP$L_SSPWA(R9)-    ;  in DAP control block
                        09AE                  SIZE=#DAP$K_SSPWA       ;
                        09B8
                        09B8  ;
                        09B8  ; Process the system specific menu field (optional).
                        09B8  ; Each bit set denotes that its associated field follows in the message.
                        09B8  ;
                        09B8
                        09B8          ASSUME  DAP$V_SSP_CAP+1 EQ DAP$V_SSP_FLG
                        09B8
      58   0A06'CF   9E 09B8          MOVAB   W^SSP_SUCCESS,R8        ; Specify transfer address on EOM
                        09BD          STORE_FIELD     SSP_MENU,2,K_EXT; Save system specific menu field
                        09C4          CHECK_MASKS     SSP_MEN,2       ; Validate bit options
      58   FF4D CF   9E 09CC          MOVAB   W^ERROR_FORMAT,R8       ; Specify transfer address on EOM
           5C   66   3C 09D1          MOVZWL  (R6),AP                 ; Copy menu to scratch register
                        09D4  SSP_LOOP:
 50   5C   02   00   EA 09D4          FFS     #0,#DAP$V_SSP_FLG+1,AP,R0; Get position of next bit set
                        09D9          $CLRBIT R0,AP                   ; Clear menu bit just found
              F4 AF  9F 09DD          PUSHAB  B^SSP_LOOP              ; Push return address on stack
                        09E0          $CASEB  SELECTOR=R0-            ; Next field:
                        09E0                  DISPL=<-                ;
                        09E0                          10$-            ;  SSP_CAP
                        09E0                          20$-            ;  SSP_FLG
                        09E0                  >
                1C   11 09E8          BRB     SSP_SUCCESS             ; All fields parsed
                        09EA
                        09EA  ;
                        09EA  ; Process each field specified in the menu (optional).
                        09EA  ;
                        09EA
                        09EA  10$:    STORE_FIELD     SSP_CAP,4,K_EXT,<M_TRUNC>
                        09F1                                          ; Save system specific capabilities
                     05 09F1          RSB                             ;  field
                        09F2  20$:    STORE_FIELD     SSP_FLG,4,K_EXT ; Save system specific flags field
                        09F9          CHECK_MASKS     SSP_FLG,4       ; Validate bit options
                     05 0A05          RSB                             ;
                        0A06
                        0A06  ;
                        0A06  ; System specific mini-message has been parsed successfully!
                        0A06  ;
                        0A06
                        0A06  SSP_SUCCESS:                            ;
                50 8ED0 0A06          POPL    R0                      ; Throw away return address on stack
```

```
                        5A 8ED0  0A09   2081          POPL    R10                      ; Restore address of end-of-message + 1
                   FF4C    31    0A0C   2082          BRW     EXIT_COMMON              ; Exit here because this routine
                                 0A0F   2083                                           ;  was entered from EXIT_SUCCESS
                                 0A0F   2084
                                 0A0F   2085          .END                             ; End of module
```

```
$$COUNT                         = 00000002                 DAP$B_NOK                           00000044
ACC_INVALID                       00000403 R       02      DAP$B_NOR                           00000046
ACC_MSG                           00000381 R       02      DAP$B_NSG                           00000049
ALL_INVALID                       00000713 R       02      DAP$B_NUL                           0000006D
ALL_LOOP                          000006A3 R       02      DAP$B_ORG                           00000045
ALL_MSG                           00000687 R       02      DAP$B_OSTYPE                        00000042
ATT_INVALID                       0000037E R       02      DAP$B_RAC                           00000046
ATT_LOOP                          00000268 R       02      DAP$B_RAT                           00000047
ATT_MSG                           0000022E R       02      DAP$B_REF                           0000006C
CHECK_FILE_SYSTEM                 000001A7 R       02      DAP$B_RFM                           00000046
CHECK_MASKS                       0000096F R       02      DAP$B_SHR                           00000043
CHECK_OPERATING_SYSTEM            000001D2 R       02      DAP$B_SIZ                           0000005C
CHECK_PROTOCOL_VERSION            00000132 R       02      DAP$B_SIZ_TMP                       0000004A
CMP_INVALID                       00000520 R       02      DAP$B_STREAMID                      00000032
CMP_MSG                           000004E5 R       02      DAP$B_TKS                           0000007F
CNF_MSG                           000000EE R       02      DAP$B_TYPE                          00000030
CON_INVALID                       000004E2 R       02      DAP$B_USRNUM                        00000046
CON_MSG                           000004CA R       02      DAP$B_USRVER                        00000048
CTL_INVALID                       000004C7 R       02      DAP$B_VERNUM                        00000044
CTL_LOOP                          0000045D R       02      DAP$B_X_FIELD                       00000024
CTL_MSG                           00000406 R       02      DAP$C_BCN                           000000C0
DAP$B_ACCFUNC                     00000040                 DAP$K_ABORT                       = 00000003
DAP$B_ACCOPT                      00000041                 DAP$K_ACCOPT_I                    = 000000E0
DAP$B_AID                         00000050                 DAP$K_ACCOPT_U                    = 00000016
DAP$B_ALN                         00000044                 DAP$K_ACC_MSG                     = 00000003
DAP$B_AOP                         00000045                 DAP$K_ACK_MSG                     = 00000006
DAP$B_BITCNT                      00000035                 DAP$K_ALLMENU_I                   = 0000FE10
DAP$B_BKS                         00000050                 DAP$K_ALLMENU_U                   = 00000000
DAP$B_BKZ                         00000051                 DAP$K_ALL_MSG                     = 0000000B
DAP$B_BLKCNT                      00000056                 DAP$K_ANY                         = 00000000
DAP$B_BSZ                         00000052                 DAP$K_AOP_I                       = 000000F0
DAP$B_CMPFUNC                     00000040                 DAP$K_AOP_U                       = 00000000
DAP$B_CONFUNC                     00000040                 DAP$K_ATTMENU_I                   = FFE08000
DAP$B_CTLFUNC                     00000040                 DAP$K_ATTMENU_U                   = 00000000
DAP$B_DAN                         00000070                 DAP$K_ATT_MSG                     = 00000002
DAP$B_DATATYPE                    00000044                 DAP$K_BLK_FILE                    = 00000005
DAP$B_DBS                         0000007C                 DAP$K_BLK_VBN                     = 00000004
DAP$B_DCODE_FID                   00000019                 DAP$K_BLN                           000000C0
DAP$B_DCODE_MAC                   0000001B                 DAP$K_BLS_D                       = 00000200
DAP$B_DCODE_MSG                   0000001A                 DAP$K_BN2                         = 00000002
DAP$B_DECVER                      00000047                 DAP$K_BN4                         = 00000004
DAP$B_DTP                         00000071                 DAP$K_BN8                         = 00000007
DAP$B_ECONUM                      00000045                 DAP$K_BSZ_D                       = 00000008
DAP$B_FAC                         00000042                 DAP$K_CHANGE_B                    = 00000006
DAP$B_FILESYS                     00000043                 DAP$K_CHANGE_E                    = 00000007
DAP$B_FLAGS                       00000031                 DAP$K_CLOSE                       = 00000001
DAP$B_FLG                         00000048                 DAP$K_CMP_MSG                     = 00000007
DAP$B_FSZ                         00000051                 DAP$K_CMWA                        = 00000050
DAP$B_IAN                         0000006E                 DAP$K_CNF_MSG                     = 00000001
DAP$B_IBS                         0000007D                 DAP$K_CONNECT                     = 00000002
DAP$B_KRF                         00000047                 DAP$K_CON_MSG                     = 00000005
DAP$B_LAN                         0000006F                 DAP$K_COPOS11                     = 0000000D
DAP$B_LEN256                      00000034                 DAP$K_CREATE                      = 00000002
DAP$B_LENGTH                      00000033                 DAP$K_CTLMENU_I                   = 0000FF90
DAP$B_LVL                         0000007E                 DAP$K_CTLMENU_U                   = 00000040
DAP$B_NAMETYPE                    00000040                 DAP$K_CTL_MSG                     = 00000004
DAP$B_NOA                         00000045                 DAP$K_CYL                         = 00000001
```

```
DAP$K_DATATYP_D                 = 00000002          DAP$K_RELEASE                   = 00000009
DAP$K_DATATYP_I                 = 00000004          DAP$K_RENAME                    = 00000003
DAP$K_DATATYP_U                 = 00000088          DAP$K_RESET                     = 00000003
DAP$K_DAT_MSG                   = 00000008          DAP$K_RESPONSE                  = 00000002
DAP$K_DELETE                    = 00000005          DAP$K_RESUME                    = 00000004
DAP$K_DEV_I                     = FC000040          DAP$K_RETRY                     = 00000001
DAP$K_DEV_U                     = 00000000          DAP$K_REWIND                    = 00000006
DAP$K_DIR_LIST                  = 00000006          DAP$K_RFA_ACC                   = 00000002
DAP$K_DISCONN                   = 00000004          DAP$K_RFI                       = 00000004
DAP$K_DISPLAY                   = 00000010          DAP$K_RFM_D                     = 00000001
DAP$K_DISPLAY_I                 = 0000FCC0          DAP$K_RMST1                     = 00000001
DAP$K_DISPLAY_U                 = 00000200          DAP$K_RMS20                     = 00000002
DAP$K_ERASE                     = 00000004          DAP$K_RMS32                     = 00000003
DAP$K_EXECUTE                   = 00000008          DAP$K_RMS32S                    = 0000000A
DAP$K_EXTEND_B                  = 0000000B          DAP$K_ROP_I                     = FFF80008
DAP$K_EXTEND_E                  = 0000000F          DAP$K_ROP_U                     = 00000000
DAP$K_FAC_D                     = 00000002          DAP$K_RSTS                      = 00000002
DAP$K_FAC_I                     = 00000000          DAP$K_RSX11D                    = 00000005
DAP$K_FAC_U                     = 00000000          DAP$K_RSX11M                    = 00000004
DAP$K_FCST1                     = 00000004          DAP$K_RSX11MP                   = 0000000C
DAP$K_FIND                      = 0000000E          DAP$K_RSX11S                    = 00000003
DAP$K_FIX                       = 00000001          DAP$K_RT11                      = 00000001
DAP$K_FLAGS_I                   = 00000090          DAP$K_RT11FS                    = 00000005
DAP$K_FLAGS_U                   = 00000048          DAP$K_SEQ                       = 00000000
DAP$K_FLG_I                     = 000000F8          DAP$K_SEQ_ACC                   = 00000000
DAP$K_FLG_U                     = 00000000          DAP$K_SEQ_FILE                  = 00000003
DAP$K_FLUSH                     = 0000000C          DAP$K_SHR_D                     = 00000000
DAP$K_FOP_I                     = F1021004          DAP$K_SHR_I                     = 00000080
DAP$K_FOP_U                     = 00002000          DAP$K_SHR_U                     = 00000010
DAP$K_FREE                      = 0000000A          DAP$K_SKIP_FILE                 = 00000005
DAP$K_GET_READ                  = 00000001          DAP$K_SKIP_REC                  = 00000002
DAP$K_IAS                       = 00000006          DAP$K_SPACE_BW                  = 00000012
DAP$K_IDX                       = 00000020          DAP$K_SPACE_FW                  = 00000011
DAP$K_IN2                       = 00000001          DAP$K_SSPWA                     = 00000010
DAP$K_IN4                       = 00000003          DAP$K_SSP_FLG_I                 = FFFFFFFE
DAP$K_IN8                       = 00000006          DAP$K_SSP_FLG_U                 = 00000000
DAP$K_KEYMENU_I                 = FFF81000          DAP$K_SSP_MEN_I                 = 0000FFFC
DAP$K_KEYMENU_U                 = 00000000          DAP$K_SSP_MEN_U                 = 00000000
DAP$K_KEY_ACC                   = 00000001          DAP$K_STG                       = 00000000
DAP$K_KEY_MSG                   = 0000000A          DAP$K_STM                       = 00000004
DAP$K_LBN                       = 00000002          DAP$K_STMCR                     = 00000006
DAP$K_NAM_MSG                   = 0000000F          DAP$K_STMLF                     = 00000005
DAP$K_NO_FS                     = 00000006          DAP$K_STS_MSG                   = 00000009
DAP$K_OPEN                      = 00000001          DAP$K_SUBMIT                    = 00000007
DAP$K_ORG_D                     = 00000000          DAP$K_SUM_MSG                   = 0000000C
DAP$K_PAC                       = 00000005          DAP$K_TEMP                      = 00000010
DAP$K_PROMENU_I                 = 0000FFE0          DAP$K_TERMINATE                 = 00000008
DAP$K_PROMENU_U                 = 00000000          DAP$K_TIMENU_I                  = 0000FF80
DAP$K_PROTECT_I                 = 0000FE00          DAP$K_TIMENU_U                  = 00000000
DAP$K_PROTECT_U                 = 00000000          DAP$K_TIM_MSG                   = 0000000D
DAP$K_PRO_MSG                   = 0000000E          DAP$K_TOPS10                    = 00000009
DAP$K_PUT_WRITE                 = 00000004          DAP$K_TOPS10FS                  = 00000008
DAP$K_P_OS                      = 0000000E          DAP$K_TOPS20                    = 00000008
DAP$K_QUIT                      = 00000005          DAP$K_TOPS20FS                  = 00000007
DAP$K_RAT_I                     = 00000020          DAP$K_TRUNCATE                  = 00000007
DAP$K_RAT_U                     = 000000C0          DAP$K_UDF                       = 00000000
DAP$K_REL                       = 00000010          DAP$K_UPDATE                    = 00000003
```

```
DAP$K_VAR                             = 00000002            DAP$Q_EDT                            00000058
DAP$K_VAXELAN                         = 0000000F            DAP$Q_FILEDATA                       00000044
DAP$K_VAXVMS                          = 00000007            DAP$Q_FILESPEC                       00000044
DAP$K_VBN                             = 00000003            DAP$Q_KEY                            00000048
DAP$K_VFC                             = 00000003            DAP$Q_KNM                            00000064
DAP$L_ALQ1                              0000004C            DAP$Q_MSG_BUF1                       00000008
DAP$L_ALQ2                              0000004C            DAP$Q_MSG_BUF2                       00000010
DAP$L_ATTMENU                           00000040            DAP$Q_NAMESPEC                       00000044
DAP$L_CMWA                              00000030            DAP$Q_OWNER                          00000048
DAP$L_CRC_RSLT                          00000020            DAP$Q_PASSWORD                       00000050
DAP$L_DCODE_STS                         00000018            DAP$Q_PDT                            00000068
DAP$L_DEV                               00000068            DAP$Q_RDT                            00000050
DAP$L_DVB                               00000078            DAP$Q_RUNSYS                         0000005C
DAP$L_EBK                               00000078            DAP$Q_STX                            00000050
DAP$L_FOP1                              00000064            DAP$Q_SYSCAP                         00000028
DAP$L_FOP2                              00000044            DAP$Q_SYSPEC                         00000038
DAP$L_HBK                               00000074            DAP$V_ADT                          = 00000006
DAP$L_KEYMENU                           00000040            DAP$V_AID                          = 00000006
DAP$L_LOC                               00000048            DAP$V_ALN                          = 00000001
DAP$L_MRN                               00000058            DAP$V_ALQ1                         = 00000006
DAP$L_MSG_MASK                          0000001C            DAP$V_ALQ2                         = 00000005
DAP$L_RECNUM1                           00000040            DAP$V_AOP                          = 00000002
DAP$L_RECNUM2                           00000048            DAP$V_BDT                          = 00000004
DAP$L_ROP                               00000050            DAP$V_BITCNT                       = 00000003
DAP$L_RVB                               00000074            DAP$V_BKS                          = 00000007
DAP$L_SBN                               0000007C            DAP$V_BKZ                          = 00000007
DAP$L_SSPWA                             00000080            DAP$V_BLKCNT                       = 00000006
DAP$L_SSP_CAP                           00000088            DAP$V_BLS                          = 00000004
DAP$L_SSP_FLG                           00000084            DAP$V_BSZ                          = 0000000D
DAP$L_STV                               0000004C            DAP$V_CDT                          = 00000000
DAP$L_TEMP                              00000090            DAP$V_DAN                          = 00000009
DAP$M_BITCNT                          = 00000008            DAP$V_DATATYPE                     = 00000000
DAP$M_BLKCNT                          = 00000040            DAP$V_DBS                          = 0000000E
DAP$M_CMPFMT                          = 00000008            DAP$V_DEQ1                         = 0000000B
DAP$M_DFTSPEC                         = 00000010            DAP$V_DEQ2                         = 00000008
DAP$M_DMO                             = 00002000            DAP$V_DEV                          = 0000000E
DAP$M_DSP_3NAM                        = 00000200            DAP$V_DFL                          = 00000001
DAP$M_DSP_ATT                         = 00000001            DAP$V_DISPLAY2                     = 00000005
DAP$M_EMBEDDED                        = 00000010            DAP$V_DTP                          = 0000000A
DAP$M_GET                             = 00000002            DAP$V_DVB                          = 0000000D
DAP$M_GO_NOGO                         = 00000010            DAP$V_EBK                          = 00000012
DAP$M_IMAGE                           = 00000002            DAP$V_EDT                          = 00000002
DAP$M_LOADIM                          = 00000001            DAP$V_FCS                          = 00000031
DAP$M_LSA                             = 00000040            DAP$V_FFB                          = 00000013
DAP$M_MACY11                          = 00000080            DAP$V_FLG                          = 00000000
DAP$M_MSE                             = 00000010            DAP$V_FOP1                         = 0000000C
DAP$M_SEGMENT                         = 00000040            DAP$V_FSZ                          = 00000008
DAP$M_TMP1$                           = 00000020            DAP$V_GEQ_V41                      = 00000020
DAP$M_TMP2$                           = 000000C0            DAP$V_GEQ_V42                      = 00000021
DAP$M_TMP3$                           = 00020000            DAP$V_GEQ_V52                      = 00000022
DAP$M_TMP4$                           = 01000000            DAP$V_GEQ_V54                      = 00000023
DAP$M_TMP5$                           = F0000000            DAP$V_GEQ_V56                      = 00000024
DAP$M_ZERO                            = 00000080            DAP$V_GEQ_V60                      = 00000025
DAP$Q_ADT                               00000070            DAP$V_GEQ_V70                      = 00000026
DAP$Q_BDT                               00000060            DAP$V_GEQ_V71                      = 00000027
DAP$Q_CDT                               00000048            DAP$V_HBK                          = 00000011
DAP$Q_DCODE_FLG                         00000000            DAP$V_IAN                          = 00000007
```

```
DAP$V_IAS                         = 0000003B                DAP$W_CHECK                         00000042
DAP$V_IBS                         = 0000000F                DAP$W_CTLMENU                       00000044
DAP$V_IFL                         = 00000002                DAP$W_DEQ1                          00000054
DAP$V_KEY                         = 00000001                DAP$W_DEQ2                          00000052
DAP$V_KNM                         = 00000005                DAP$W_DFL                           00000044
DAP$V_KRF                         = 00000002                DAP$W_DISPLAY1                      0000004C
DAP$V_LAN                         = 00000008                DAP$W_DISPLAY2                      00000054
DAP$V_LEN256                      = 00000002                DAP$W_FFB                           00000072
DAP$V_LENGTH                      = 00000001                DAP$W_IFL                           00000046
DAP$V_LOC                         = 00000003                DAP$W_LRL                           00000070
DAP$V_LRL                         = 00000010                DAP$W_MRL                           00000072
DAP$V_LVL                         = 00000010                DAP$W_MRS                           0000004A
DAP$V_MRL                         = 00000012                DAP$W_PARTNER                       00000006
DAP$V_MRN                         = 00000009                DAP$W_POS                           0000004C
DAP$V_MRS                         = 00000005                DAP$W_POS_TMP                       0000004A
DAP$V_NSG                         = 00000003                DAP$W_PROGRP                        00000054
DAP$V_NUL                         = 00000006                DAP$W_PROMENU                       00000040
DAP$V_ORG                         = 00000001                DAP$W_PROOWN                        00000052
DAP$V_OWNER                       = 00000000                DAP$W_PROSYS                        00000050
DAP$V_PDT                         = 00000005                DAP$W_PROWLD                        00000056
DAP$V_PROGRP                      = 00000003                DAP$W_PVN                           00000042
DAP$V_PROOWN                      = 00000002                DAP$W_RFA                           00000042
DAP$V_PROSYS                      = 00000001                DAP$W_RVN                           00000042
DAP$V_PROWLD                      = 00000004                DAP$W_SSP_MENU                      00000080
DAP$V_P_OS                        = 0000003C                DAP$W_STSCODE                       00000040
DAP$V_RAC                         = 00000000                DAP$W_SUMENU                        00000040
DAP$V_RAT                         = 00000003                DAP$W_TIMENU                        00000040
DAP$V_RDT                         = 00000001                DAP$W_VERSION                       00000004
DAP$V_REF                         = 00000004                DAP$W_VOL                           00000042
DAP$V_RFM                         = 00000002                DAP$_ACCFUNC                      = 00000010
DAP$V_RMS                         = 00000030                DAP$_ACCOPT                       = 00000011
DAP$V_ROP                         = 00000003                DAP$_ADT                          = 00000017
DAP$V_RSTS                        = 00000039                DAP$_AID                          = 00000017
DAP$V_RSX                         = 0000003A                DAP$_ALLMENU                      = 00000010
DAP$V_RT11                        = 00000038                DAP$_ALN                          = 00000012
DAP$V_RUNSYS                      = 0000000A                DAP$_ALQ1                         = 00000017
DAP$V_RVB                         = 0000000B                DAP$_ALQ2                         = 00000016
DAP$V_RVN                         = 00000003                DAP$_AOP                          = 00000013
DAP$V_SBN                         = 00000014                DAP$_ATTMENU                      = 00000010
DAP$V_SEGMENT                     = 00000006                DAP$_BDT                          = 00000015
DAP$V_SSP_CAP                     = 00000000                DAP$_BKS                          = 00000018
DAP$V_SSP_FLG                     = 00000001                DAP$_BKZ                          = 00000018
DAP$V_STM_ONLY                    = 00000032                DAP$_BLKCNT                       = 00000018
DAP$V_STREAMID                    = 00000000                DAP$_BLS                          = 00000015
DAP$V_SYSPEC                      = 00000005                DAP$_BSZ                          = 0000001E
DAP$V_TKS                         = 00000011                DAP$_BUFSIZ                       = 00000010
DAP$V_TOPS10                      = 00000036                DAP$_CDT                          = 00000011
DAP$V_TOPS20                      = 00000037                DAP$_CHECK                        = 00000012
DAP$V_VAXELAN                     = 00000035                DAP$_CMPFUNC                      = 00000010
DAP$V_VAXVMS                      = 00000034                DAP$_CONFUNC                      = 00000010
DAP$V_VMS_XPF1                    = 0000002C                DAP$_CTLFUNC                      = 00000010
DAP$V_VOL                         = 00000000                DAP$_CTLMENU                      = 00000011
DAP$V_X_CHECK                     = 00000001                DAP$_DAN                          = 0000001C
DAP$V_X_RECNUM                    = 00000000                DAP$_DATATYPE                     = 00000011
DAP$W_ACLMENU                       00000040                DAP$_DBS                          = 00000021
DAP$W_BLS                           00000048                DAP$_DECVER                       = 00000016
DAP$W_BUFSIZ                        00000040                DAP$_DEQ1                         = 0000001C
```

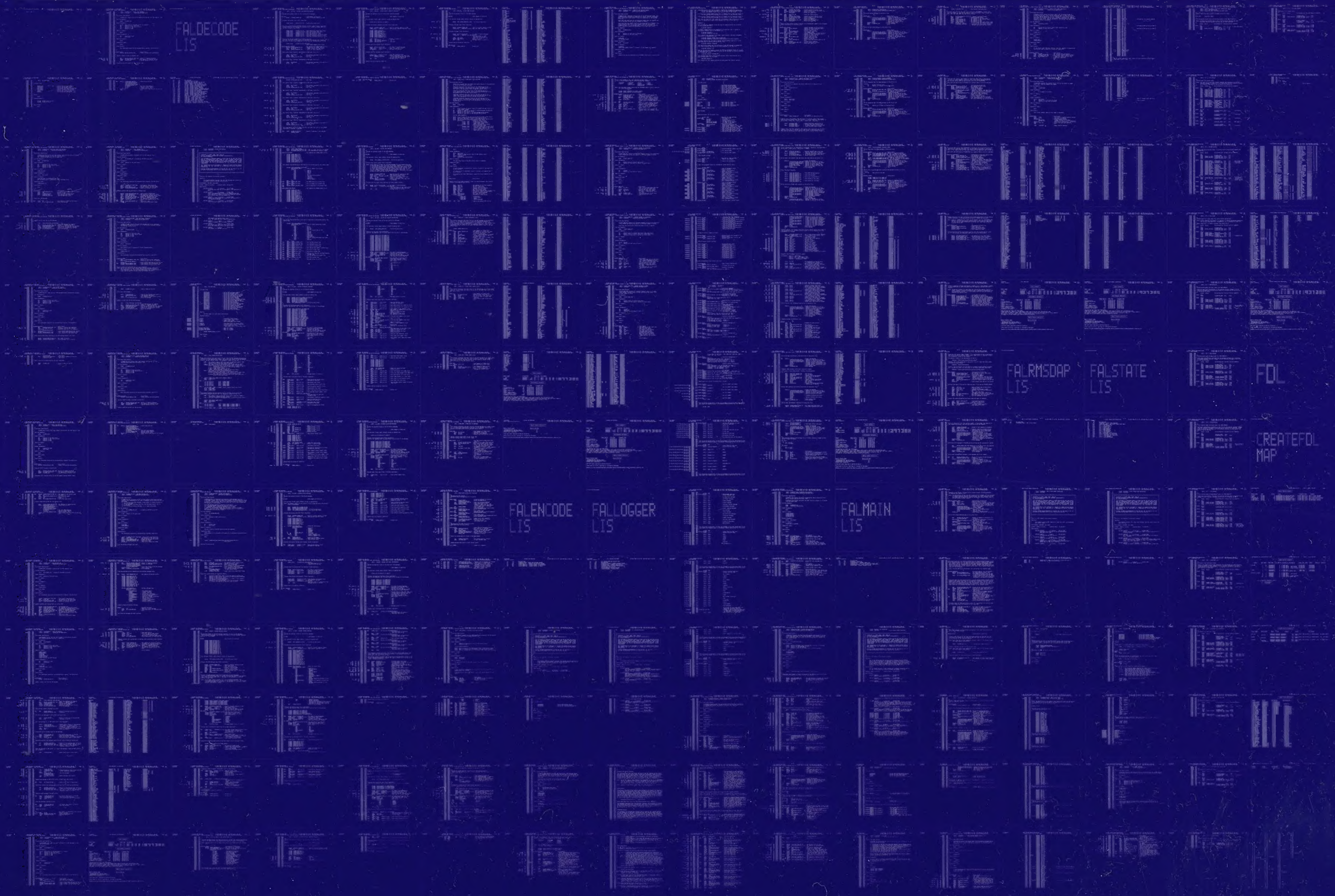
```
DAP$_DEQ2                    = 00000019          DAP$_RDT                     = 00000012
DAP$_DEV                     = 0000001F          DAP$_RECNUM1                 = 00000010
DAP$_DFL                     = 00000012          DAP$_REF                     = 00000017
DAP$_DISPLAY1                = 00000015          DAP$_RFM                     = 00000013
DAP$_DISPLAY2                = 00000017          DAP$_ROP                     = 00000015
DAP$_DTP                     = 0000001D          DAP$_RUNSYS                  = 0000001B
DAP$_DVB                     = 00000020          DAP$_RVB                     = 0000001E
DAP$_EBK                     = 00000023          DAP$_RVN                     = 00000014
DAP$_ECONUM                  = 00000014          DAP$_SBN                     = 00000025
DAP$_EDT                     = 00000013          DAP$_SHR                     = 00000014
DAP$_FAC                     = 00000013          DAP$_SIZ_TMP                 = 00000016
DAP$_FFB                     = 00000024          DAP$_SSP_CAP                 = 0000000E
DAP$_FILEDATA                = 00000011          DAP$_SSP_FLG                 = 0000000E
DAP$_FILESPEC                = 00000012          DAP$_SSP_MENU                = 0000000E
DAP$_FILESYS                 = 00000012          DAP$_STREAMID                = 00000009
DAP$_FLAGS                   = 00000008          DAP$_SYSCAP                  = 00000018
DAP$_FLG                     = 00000011          DAP$_SYSPEC                  = 0000000E
DAP$_FOP1                    = 0000001D          DAP$_TIMENU                  = 00000010
DAP$_FOP2                    = 00000011          DAP$_TKS                     = 00000024
DAP$_FORMAT                  = 00000008          DAP$_TYPE                    = 00000008
DAP$_FSZ                     = 00000019          DAP$_UNKNOWN                 = 00000000
DAP$_HBK                     = 00000022          DAP$_UNSUPPORT               = 00000002
DAP$_IAN                     = 0000001A          DAP$_USRNUM                  = 00000015
DAP$_IBS                     = 00000022          DAP$_USRVER                  = 00000017
DAP$_IFL                     = 00000013          DAP$_VERNUM                  = 00000013
DAP$_INVALID                 = 00000009          DAP$_VOL                     = 00000011
DAP$_KEY                     = 00000013          DAT_MSG                        00000523 R     02
DAP$_KEYMENU                 = 00000010          DESCRIPTOR                     00000919 R     02
DAP$_KNM                     = 00000018          DISPATCH_TABLE                 000000C8 R     02
DAP$_KRF                     = 00000014          ERROR_COMMON                   0000093C R     02
DAP$_LAN                     = 0000001B          ERROR_FORMAT                   0000091D R     02
DAP$_LEN256                  = 0000000B          ERROR_INVALID                  0000092C R     02
DAP$_LENGTH                  = 0000000A          ERROR_SYNC                     00000932 R     02
DAP$_LOC                     = 00000014          ERROR_UNSUPPORT                00000938 R     02
DAP$_LRL                     = 00000021          EXIT_COMMON                    0000095B R     02
DAP$_LVL                     = 00000023          EXIT_SUCCESS                   00000945 R     02
DAP$_MRL                     = 00000025          FAL$DECODE_MSG                 00000000 RG    02
DAP$_MRN                     = 0000001A          HDR_INVALID                    000000C2 R     02
DAP$_MRS                     = 00000016          HDR_LOOP                       0000005E R     02
DAP$_MSG_SYNC                = 0000000A          HDR_UNSUPPORT                  000000C5 R     02
DAP$_NAMESPEC                = 00000011          HEADER                         00000022 R     02
DAP$_NAMETYPE                = 00000010          KEY_INVALID                    00000684 R     02
DAP$_NSG                     = 00000014          KEY_LOOP                       00000573 R     02
DAP$_NUL                     = 00000019          KEY_MSG                        0000054F R     02
DAP$_ORG                     = 00000012          K_EXT                        = 00000000
DAP$_OSTYPE                  = 00000011          K_FIX                        = 00000001
DAP$_OWNER                   = 00000011          K_IMG                        = 00000002
DAP$_PASSWORD                = 00000016          K_ROM                        = 00000003
DAP$_PDT                     = 00000016          MOVE_FIELD                     000008FD R     02
DAP$_POS_TMP                 = 00000015          M_DESC                       = 00000010
DAP$_PROGRP                  = 00000014          M_SRCR3                      = 00000040
DAP$_PROMENU                 = 00000010          M_TRUNC                      = 00000020
DAP$_PROOWN                  = 00000013          NAM_INVALID                    0000087D R     02
DAP$_PROSYS                  = 00000012          NAM_MSG                        00000859 R     02
DAP$_PROWLD                  = 00000015          PRO_INVALID                    00000856 R     02
DAP$_RAC                     = 00000012          PRO_LOOP                       00000801 R     02
DAP$_RAT                     = 00000014          PRO_MSG                        000007E1 R     02
```

```
SSP_LOOP                          000009D4 R      02
SSP_MINI_MSG                      000009A5 R      02
SSP_SUCCESS                       00000A06 R      02
STORE_EXT                         000008AD R      02
STORE_FIELD                       00000880 R      02
STORE_FIX                         000008D5 R      02
STORE_IMG                         000008E5 R      02
STORE_IMG1                        000008E8 R      02
STORE_ROM                         000008F3 R      02
SYS$BINTIM                        ********   GX   02
TIM_INVALID                       000007DE R      02
TIM_LOOP                          00000732 R      02
TIM_MSG                           00000716 R      02
TMPT..                          = 000009FD R      02
TMP2..                          = 00000A05 R      02
V_DESC                          = 00000004
V_SRCR3                         = 00000006
V_TRUNC                         = 00000005

                                                  +-----------------+
                                                  ! Psect synopsis !
                                                  +-----------------+

PSECT name                        Allocation          PSECT No.  Attributes
----------                        ----------          ---------  ----------
.  ABS  .                         00000000  (     0.)  00 (   0.)  NOPIC    USR    CON    ABS    LCL NOSHR NOEXE NORD   NOWRT NOVEC BYTE
$ABS$                             000000C0  (   192.)  01 (   1.)  NOPIC    USR    CON    ABS    LCL NOSHR   EXE   RD     WRT NOVEC BYTE
FAL$CODE                          00000A0F  ( 2575.)   02 (   2.)  NOPIC    USR    CON    REL    LCL NOSHR   EXE   RD   NOWRT NOVEC BYTE

                                                +--------------------------+
                                                ! Performance indicators !
                                                +--------------------------+

Phase                        Page faults    CPU Time        Elapsed Time
-----                        -----------    --------        ------------
Initialization                        35    00:00:00.04     00:00:01.80
Command processing                   132    00:00:00.36     00:00:02.84
Pass 1                               454    00:00:16.12     00:01:05.58
Symbol table sort                      0    00:00:01.10     00:00:04.06
Pass 2                               355    00:00:04.09     00:00:12.98
Symbol table output                   51    00:00:00.30     00:00:01.06
Psect synopsis output                  0    00:00:00.02     00:00:00.02
Cross-reference output                 0    00:00:00.00     00:00:00.00
Assembler run totals                1029    00:00:22.04     00:01:28.35

The working set limit was 2250 pages.
124235 bytes (243 pages) of virtual memory were used to buffer the intermediate code.
There were 60 pages of symbol table space allocated to hold 1010 non-local and 156 local symbols.
2085 source lines were read in Pass 1, producing 24 object records in Pass 2.
36 pages of virtual memory were used to define 35 macros.
```

```
                                    +----------------------------+
                                    ! Macro library statistics !
                                    +----------------------------+

Macro library name                  Macros defined
------------------                  --------------
_$255$DUA28:[FAL.OBJ]FAL.MLB;1            24
_$255$DUA28:[SYSLIB]STARLET.MLB;2          6
TOTALS (all libraries)                    30

1460 GETS were required to define 30 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS$:FALDECODE/OBJ=OBJ$:FALDECODE MSRC$:FALDECODE/UPDATE=(ENH$:FALDECODE)+LIB$:FAL/LIB
```

0175 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY



FALDECODE
LIS

FALRMSDAP
LIS

FALSTATE
LIS

FDL

CREATEFDL
MAP

FALENCODE
LIS

FALLOGGER
LIS

FALMAIN
LIS